Titre: Title:	Compressive Sensing and Multichannel Spike Detection for Neuro-Recording Systems
Auteur: Author:	Nan Li
Date:	2016
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Li, N. (2016). Compressive Sensing and Multichannel Spike Detection for Neuro-Recording Systems [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2130/

UNIVERSITÉ DE MONTRÉAL

COMPRESSIVE SENSING AND MULTICHANNEL SPIKE DETECTION FOR NEURO-RECORDING SYSTEMS

NAN LI

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIAE DOCTOR

(GÉNIE ÉLECTRIQUE)

AVRIL 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

COMPRESSIVE SENSING AND MULTICHANNEL SPIKE DETECTION FOR NEURO-RECORDING SYSTEMS

présentée par : LI Nan

en vue de l'obtention du diplôme de : Philosophiae Doctor

a été dûment acceptée par le jury d'examen constitué de :

M. DAVID Jean-Pierre, Ph.D., président

M. SAWAN Mohamad, Ph.D., membre et directeur de recherche

M. ZHU Guchuan, Doctorat, membre

M. BOUKADOUM Mounir, Ph.D., membre externe

DEDICATION

To My Beloved Family and Friends

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor, Professor Mohamad Sawan for all his support, encouragement and guidance during my Ph.D. research life in Polystim Neurotechnologies Laboratory at Polytechnique Montreal.

I would also like to thank Hicham Semmaoui and Yushan Zheng for their suggestions at the early stage of my Ph.D. research. Their resourceful experience helped me get rid of some unnecessary detours in research.

Thanks are due to four intern students I guided, Rebai Hassen, Fatma Hawary, Ousamah Younoss Soliman and Morgan Osborn, for their hard work to contribute to my research project.

I thank all staff members and colleagues of Polystim Neurotech Lab who helped me during my stay in Polytechnique Montreal. Special thanks are due to the following people for their helps and collaborations: Marie-Yannick Laplante, Rejean Lapage, Jean Bouchard, Arash Moradi, Sami Hached, Zied Koubaa, Md Hasanuzzaman, Bahareh Ghane-Motlagh, Mohamed Zgaren, and Ghazal Nabovati.

I am also grateful for the support from the Canada Research Chair in Smart Medical Devices, the Natural Sciences and Engineering Research Council of Canada, CMC Microsystems and the scholarship from China Scholarship Council.

Finally, I want to express the deepest gratitude to my family for their love and encouragements during my study.

RÉSUMÉ

Les interfaces cerveau-machines (ICM) sont de plus en plus importantes dans la recherche biomédicale et ses applications, tels que les tests et analyses médicaux en laboratoire, la cérébrologie et le traitement des dysfonctions neuromusculaires. Les ICM en général et les dispositifs d'enregistrement neuronaux, en particulier, dépendent fortement des méthodes de traitement de signaux utilisées pour fournir aux utilisateurs des renseignements sur l'état de diverses fonctions du cerveau. Les dispositifs d'enregistrement neuronaux courants intègrent de nombreux canaux parallèles produisant ainsi une énorme quantité de données. Celles-ci sont difficiles à transmettre, peuvent manquer une information précieuse des signaux enregistrés et limitent la capacité de traitement sur puce. Une amélioration de fonctions de traitement du signal est nécessaire pour s'assurer que les dispositifs d'enregistrements neuronaux peuvent faire face à l'augmentation rapide des exigences de taille de données et de précision requise de traitement.

Cette thèse regroupe deux approches principales de traitement du signal - la compression et la réduction de données - pour les dispositifs d'enregistrement neuronaux. Tout d'abord, l'échantillonnage comprimé (AC) pour la compression du signal neuronal a été utilisé. Ceci implique l'usage d'une matrice de mesure déterministe basée sur un partitionnement selon le minimum de la distance Euclidienne ou celle de la distance de Manhattan (MDC). Nous avons comprimé les signaux neuronaux clairsemmés (Sparse) et non-clairsemmés et les avons reconstruit avec une marge d'erreur minimale en utilisant la matrice MDC construite plutôt.

La réduction de données provenant de signaux neuronaux requiert la détection et le classement de potentiels d'actions (PA, ou spikes) lesquelles étaient réalisées en se servant de la méthode d'appariement de formes (templates) avec l'inférence bayésienne (Bayesian inference based template matching - BBTM). Par comparaison avec les méthodes fondées sur l'amplitude, sur le niveau d'énergie ou sur l'appariement de formes, la BBTM a une haute précision de détection, en particulier pour les signaux à faible rapport signal-bruit et peut séparer les potentiels d'actions reçus à partir des différents neurones et qui chevauchent. Ainsi, la BBTM peut automatiquement produire les appariements de formes nécessaires avec une complexité de calculs relativement faible.

Enfin, nous avons complété la mise en œuvre d'un système numérique adaptatif de signaux neuronaux en temps réel, regroupant un détecteur de PA et un compresseur de données basé sur

la technique d’échantiollannage compressé. Nous avons validé les conceptions d’un seul canal et des multicanaux. Comparé aux systèmes actuels d’enregistrement de signaux neuronaux, le système proposé peut efficacement comprimer un grand nombre d’échantillons acquis et reconstruire les signaux originaux avec une petite erreur; en outre, il présente une faible consommation de puissance et possède une petite surface de silicium. Le prototype du système est prometteur pour l'application dans une large gamme d'interfaces d'enregistrement neuronales.

ABSTRACT

Brain-Machine Interfaces (BMIs) are increasingly important in biomedical research and health care applications, such as medical laboratory tests and analyses, cerebrology, and complementary treatment of neuromuscular disorders. BMIs, and neural recording devices in particular, rely heavily on signal processing methods to provide users with information. Current neural recording devices integrate many parallel channels, which produce a huge amount of data that is difficult to transmit, cannot guarantee the quality of the recorded signals and may limit on-chip signal processing capabilities. An improved signal processing system is needed to ensure that neural recording devices can cope with rapidly increasing data size and accuracy requirements.

This thesis focused on two signal processing approaches – signal compression and reduction – for neural recording devices. First, compressed sensing (CS) was employed for neural signal compression, using a minimum Euclidean or Manhattan distance cluster-based (MDC) deterministic sensing matrix. Sparse and non-sparse neural signals were substantially compressed and later reconstructed with minimal error using the built MDC matrix. Neural signal reduction required spike detection and sorting, which was conducted using a Bayesian inference-based template matching (BBTM) method. Compared with amplitude-based, energy-based, and some other template matching methods, BBTM has high detection accuracy, especially for low signal-to-noise ratio signals, and can separate overlapping spikes acquired from different neurons. In addition, BBTM can automatically generate the needed templates with relatively low system complexity. Finally, a digital online adaptive neural signal processing system, including spike detector and CS-based compressor, was designed. Both single and multi-channel solutions were implemented and evaluated. Compared with the signal processing systems in current use, the proposed signal processing system can efficiently compress a large number of sampled data and recover original signals with a small reconstruction error; also it has low power consumption and a small silicon area. The completed prototype shows considerable promise for application in a wide range of neural recording interfaces.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC	Analog-to-Digital convertor
AFPR	Average false positive rate
AP	Action potential
ASP	Analog signal processing
ATPR	Average true positive rate
BBTM	Bayesian inference based template matching
BCH	Bose-Chaudhuri-Hocquenghem
BMI	Brain machines interface
BP	Basis pursuit
BSBL	Block sparse Bayesian learning algorithm
CEER	Compression error of the expected measurement
CER	Compression error
ChS	Channel-to-scan parameter
CMOS	Complementary metal-oxide semiconductor
CS	Compressed sensing
CR	Compression rate
CRIP	Cluster restricted isometry property
DD	Discrete derivatives
DFT	Discrete Fourier transmission
DSP	Digital signal processing
EC-PC	Exponential component–polynomial component
ECG	Electrocardiography
ECoG	Electrocorticography
EEG	Electroencephalography
FM	Frequency modulating
fMRI	Functional magnetic resonance imaging
FPGA	Field programmable gate array

FPR	False positive rate
FSDE	First and second derivative extrema
FSK	Frequency shift keying
GLRTs	Generalized likelihood ratio test detection
IRLS	Iterative reweighted least square algorithm
Lasso	Least absolute shrinkage and selection operator
LDPC	Low-density parity-check
LRT	Likelihood ratio test detection
MAD	Median absolute deviation
MD	Maximum distance
MDC	Minimum Euclidean or Manhattan distance cluster-based deterministic
MEG	Magnetoencephalography
MMS	Maximum minimum spread sorting
mMMS	Modified maximum and minimum spread estimation method
MP	Matching pursuit
NEO	Nonlinear energy operator
NIRS	Near infrared spectroscopy
NMDC	Normalized MDC
OMP	Orthogonal matching pursuit algorithm
OTA	Operational transconductance amplifier
PRBS	Pseudo-random bit sequence
RER	Reconstruction error
RF	Radio frequency
RIP	Restricted isometry property
RMS	Root mean square method
SAR	Successive approximation register
SD	Standard deviation
SNR	Signal-to-noise ratio

SPI	Standard series peripheral interface
SR	Scan rate
STEO	Modified smooth teager energy operator
StOMP	Stagewise orthogonal matching pursuit algorithm
SWT	Stationary wavelet transform
S_STEO	Standard smooth teager energy operator
TCP	Threshold control parameter
TEO	Teager energy operator
TDM-FM	Time-division-multiplexing, frequency-modulating
TPR	True positive rate
UMDC	Unit MDC
USB	Universal serial bus
UWB	Ultra-wideband
VCD	Value change dump
VHDL	VHSIC (very-high-speed integrated circuits) hardware description language

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Research Motivation

Brain-machine interfaces (BMIs) have been the subject of a large amount of neuroscientific research since the 1970s [1]. According to the ways of the recording or manipulation, BMIs can be divided into Electroencephalography (EEG), Magnetoencephalography (MEG), Electrocorticography (ECoG), neural recording, Functional magnetic resonance imaging (fMRI), Near infrared spectroscopy (NIRS), etc. Implantable neural recording devices are an important category of the BMIs, which allow researchers to directly acquire signals from single and multiple neuron(s).

Due to the growing sophistication and data collection capacity of neuroscientific research and applications, BMIs need to integrate many functions and process increasingly large amounts of data, which causes that the signal processing becomes an indispensable part. For example, the analysis of EEG signals and fMRI requires feature extraction and classification methods [1] [2]; independent component analysis is used for analysis of MEG signals [3]; Kernel-based learning methods are used to analyse ECoG signals [4]. Designing a real-time adaptive BMI has become a hot topic [5] [6] [7].

Implantable neural recording devices are an important category of BMIs: they allow researchers to directly acquire signals from single and multiple neuron(s). However, an implantable real-time adaptive neural recording device faces many challenges. First of all, it must integrate an ever-increasing number of channels to improve recording performance. The multichannel neural recording system must provide information about neurons at multiple sites and also about the relationship between these neuronal sites. More channels means a huge amount of data must be collected, which presents difficulties in storing, processing and transmitting data to a base station. Also, an implantable device has some challenging design limitations: its surface area should be tiny; it should maintain low temperature in order to protect tissue from heat injury; and it should have low power consumption to permit a long lifetime.

Achieving the fast and accurate neural signal processing needed by an implantable real-time adaptive neural recording device is a similarly challenging goal. Current neural signal processing methods can be divided into two principal strategies: signal reduction and compression. Spike

detection and sorting are popular signal reduction methods, but despite considerable research, several difficulties remain to be overcome. The first major problem is detection accuracy. The spike detection block should correctly detect all of the spikes while removing the noise, and the detection system should have low complexity to ease its implementation. Furthermore, the spike processing system should separate the overlapping spikes that originate from different neurons.

Signal compression can keep the original signal to the maximum possible extent while reducing the burden of the transmission, and therefore it has aroused much interest among designers of neural recording devices [8] [9] [10]. A new signal compression technique called compressed sensing (CS) for use in processing recorded signals has been discussed [11] [12]; however, the neural signal is usually not sparse in the time domain, so the application of the CS technique for neural signals needs further research. Moreover, compressing neural signals requires development of a dedicated sensing matrix with a high compression rate and a low reconstruction error.

Finally, signal processing algorithms should be applied carefully for the circuit design. To date, designers have focused on the design of the front-end circuit and transmitter, but the need to design and develop a high-efficiency, low-cost signal processing system is becoming more and more pressing. The two principal difficulties hindering the development of such a system are the lack of a suitably high-performance and low-complexity signal processing technique, and the non-existence of a circuit design with sufficiently low power consumption.

1.2 Objectives

The main objective of this research was to study new approaches, both in theory and implementation, for spike detection (sorting) and signal compression in a neural recording device. The specific objectives were as follows:

- to develop a sensing matrix for the compression of neural signals using the CS technique;
- to understand the process of reconstruction of the original neural signals and determine the influence of parameters such as the sampling rate and length of the signal;
- to evaluate the high-efficiency spike detection method, including high-accuracy detection and classification of the spikes;

- to design a digital neural signal processing system that includes spike detection and signal compression, and to test and verify the proposed system.

1.3 Contributions

The contributions of this thesis and our research are summarized as follows:

- New methods about the compression of sparse and non-sparse neural signals with CS technique. A minimum Euclidean or Manhattan distance cluster-based deterministic compressed (MDC) sensing matrix was proposed to compress the neural signal. The MDC can compress sparse and non-sparse signals using the similarity, which is appropriate for the compression of neural signals in the time domain. We also give the mathematic proof of the MDC matrix for compression. Furthermore, the results of our research into other compression methods based on CS technique are outlined.
- New knowledge about the reconstruction of original neural signals with different reconstruction methods. We found that the unit MDC matrix that is composed of zero and one can be used for the compression of neural signals, which has low complexity suitable for the design of the compression system in neural recording devices. The factors that influence the MDC matrix, such as the length of signals, sampling rate, are identified and discussed.

The above contributions are detailed in the following published articles:

N. Li and M. Sawan, "Neural signal compression using a minimum Euclidean or Manhattan distance cluster-based deterministic compressed sensing matrix," *Biomedical Signal Processing and Control*, vol. 19, pp. 44-55, 2015.

H. Semmaoui, N. Li, S. Khayat-Hosseini, J. Martinez-Trujillo, and M. Sawan, "An adaptive recovery method in compressed sensing of extracellular neural recording," *Journal of Neurology and Neuroscience*, vol. 6(19), pp.1-11, 2015.

- A spike detection and sorting method with a high detection and classification accuracy was proposed; it is based on Bayesian inference-based template matching. Using this system, spikes can be detected with high accuracy, especially for a low signal-to-noise ratio (SNR). Also, the overlapping spikes can be separated and classified. Furthermore, the system can

automatically generate the templates. Finally, the proposed system has a simple structure which can be used for circuit implementation.

- An amplitude-based thresholding method of spike detection, called modified Maximum and Minimum Spread (mMMS) Estimation Method, was tested. Compared with the original MMS method, mMMS has low power consumption and good detection accuracy for high SNR.

The above contributions are reported in the following articles:

N. Li, H. Semmaoui, and M. Sawan, "Modified Maximum and Minimum Spread estimation method for detection of neural spikes," *Proceedings, 2013 IEEE International Conference on Electronics, Circuits, and Systems*, pp. 530-533.

N. Li, L. Fang and M. Sawan, "An efficient real-time neural spike detection method based on Bayesian inference with automatic template generation" (under review).

- The design of a neural signal processing system, including spike detection and signal compression, is presented. The design is divided into single-channel and multichannel systems. Based on the single-channel system, the signal processing for a 256-channel multichannel system is discussed. The implemented digital circuit is tested and verified by simulation software and the field-programmable gate array (FPGA) testing board. The proposed system has good processing performance and relatively low power consumption and a small silicon area, which can be used in the neural recording interfaces.

The details of this contribution can be found in the following articles:

N. Li, M. Osborn, G. Wang and M. Sawan, "A Digital multichannel neural signal processing system using compressed sensing technique" Accepted for publication by *Elsevier Digital Signal Processing*.

N. Li and M. Sawan, "High compression rate and efficient spikes detection system using compressed sensing technique for neural signal processing," *Proceedings, 7th International IEEE/EMBS Conference on Neural Engineering*, 2015, pp. 597-600.

N. Li, M. Osborn, L. Fang and M. Sawan, "Using Template Matching and Compressed Sensing Techniques to Enhance Performance of Spike Detection and Data Compression Systems" Accepted by 2016 IEEE International Symposium on Circuits and Systems.

1.4 Thesis Organization

This thesis is written in a paper-based format.

Chapter 2 contains a review of BMIs, neural recording systems and neural signal processing. First, it describes BMIs and their uses, and introduces several signal acquisition techniques of BMIs. Neural recording systems are specifically highlighted, and several systems and processing techniques are compared. All the significant related work in neural recording and signal processing techniques is reviewed. This chapter is one part of a review paper being prepared for submission to a high-ranking circuits and systems journal.

A neural signal compression system based on the CS technique is discussed in chapter 3, where a sensing matrix, called a minimum Euclidean or MDC sensing matrix, is introduced. This chapter explores several key points relating to this sensing matrix and proves that the proposed sensing matrix can be used for neural signal compression. This chapter is published in *Biomedical Signal Processing and Control (*vol. 19, pp. 44-55, 2015).

In Chapter 4, we propose an automatic template generation system using a Bayesian inference-based template matching method for spike detection and classification. This system accurately detects spikes and classifies spikes. The chapter describes the system and its detection and classification accuracy.

A digital online adaptive neural signal processing system, including spike detection and compression, is implemented in chapter 5. The single-channel processing system includes a spike detection block using the RMS method and a compression block using the MDC matrix. Based on the single-channel design, we investigate the signal processing of a multichannel system and present our results. Finally, the system is verified with an FPGA testing board. This chapter will be published in *Elsevier Digital Signal Processing*.

Chapter 6 contains the general discussion for the thesis, and our conclusions, along with recommendations for future work, are presented in chapter 7.

CHAPTER 2 STATE OF THE ART OF NEURAL RECORDING INTERFACES AND NEURAL SIGNAL PROCESSING TECHNIQUES

In this chapter, we begin with a discussion on BMIs, then review neural recording devices. Finally, we review neural signal processing methods, including spike detection and the CS technique for signal compression.

2.1 Brain-Machine Interfaces

Biomedical signals are important information in research on the human physiological processes. Human bodies are made up of many systems, and each system is comprised of several subsystems that carry on numerous physiological processes. These processes are complex phenomena, and their nature and activities can be reflected by various biomedical signals. These signals can be biochemical (in the form of hormones and neurotransmitters), bioelectrical (in the form of action potentials), or physical (in the form of pressure or temperature) [13].

A BMI is a system which enables the acquisition of information about cerebral activity and also permits the brain to control computers or other devices. The human body can interact with the control signals that are generated by such a system. BMIs can improve the quality of life and reduce the cost of daily care for people with restricted mobility and physical disabilities, through linking to external devices such as computers and assistive appliances which respond to patients' requirements.

The function of a typical BMI contains five consecutive stages: signal capture, preprocessing of the signal, signal processing, transmission or stimulation, and results analysis or evaluation. The signal capture stage collects biomedical signals. The preprocessing stage prepares signals to be more recognizable in order to deal with them most effectively in the following step. The signal processing stage satisfies the BMI user's specific requirements with respect to the calculation of the acquired signals, such as feature extraction or spike classification. The transmission or stimulation stage either transmits the acquired signals or stimulates organs or tissues. The final step is the analysis of the acquired signals or the evaluation of the stimulation performance.

Two main categories of the neural signals in the brain can be monitored. One is electrophysiological activity, and the other is hemodynamic activity [14]. Currently, most BMI

devices use electrophysiological activity to acquire information from the brain. This can be done through two approaches: noninvasive methods and invasive methods. The noninvasive method does not involve surgery being performed on a patient to acquire the signals, so it has minimum risk, considerable convenience for research, and makes recruiting participants much easier [15]. The invasive approach requires implantation of the device into a living body, so most invasive BMIs have been tested only in experimental animals [16]. Five approaches to communicating with the human brain – some invasive, some non-invasive –are introduced below.

Magnetoencephalography (MEG) is a noninvasive method that records the brain's magnetic activity by means of magnetic induction. MEG has the advantage that magnetic fields are rarely distorted by the skull and scalp, unlike electric fields [17]. A disadvantage of this method is the size and the high price of the acquisition [14]. In addition, the accuracy and flexibility of the MEG still needs to be improved [18] [19].

Electroencephalography (EEG) is a noninvasive method which measures the voltage fluctuations in brain activity caused by the flow of electric current due to the synaptic excitations of the dendrites of the neurons. EEG data collection occurs through electrodes placed on the scalp. Because of its simplicity, it is the most widespread neuronal recording method and has many applications; for example, it can be used to monitor epilepsy [20] [21].

Electrocorticography (ECoG) is an invasive method in which electrodes are placed directly on the surface of the brain to record the electrical activity in the region of the cerebral cortex. Compared with EEG, ECoG has good recording resolution, because it bypasses the signal-distorting skull and intermediate tissue; thus it is suitable for the study of activity such as blinks and eye movement [22] [23].

Functional Magnetic Resonance Imaging (fMRI) is a noninvasive method; it uses the electromagnetic fields to detect changes in cerebral blood flow and oxygenation levels during neural activity. fMRI is often used for blood-oxygen-level dependent contrast imaging [24], but is also used in research and treatment monitoring applications for conditions such as epilepsy and language processing deficiencies [25] [26] [27] [28].

Compared with the four signal recording methods described above, a neural recording system has great promise for advancing the understanding of brain function by allowing scientists to directly observe and analyze neural activity during normal animal behavior [29]. A neural

recording techniques can be divided into single-electrode and multi-electrode recording methods. Single-electrode neural recording was popular until the 1960s [30] [31]. A recording from a single electrode can reveal the characteristics of one or a few cell(s), but it cannot give information about how neurons networks work together to process information, which requires the use of arrays of microelectrodes to study temporal and spatial relationships between groups of neurons [32]. Therefore, single-electrode recording was eventually replaced by multi-electrode recording.

The first multi-electrode system was proposed by Marg and Adams in 1967 [33]. Since then, multi-electrode or multichannel systems have become mainstream in the neural recording field. Multichannel neural recording reveals the importance of observing the activity and interaction of many neurons simultaneously [32]. Figure 2.1 shows a typical system used to monitor and record neural signals [34]. This system includes the recording electrodes (or probes), the inner and external signal processing circuits and systems, and the wireless transceiver.

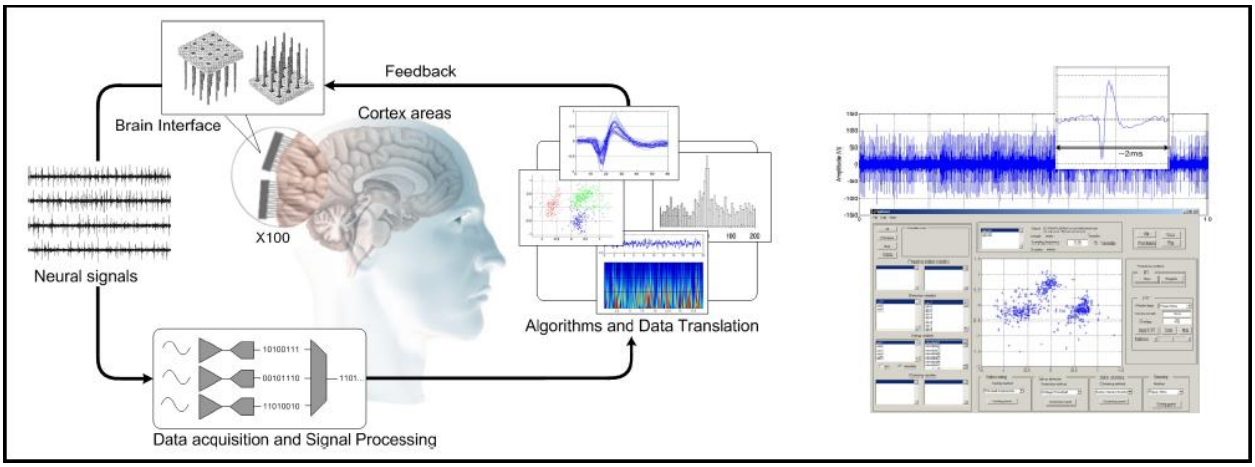


Figure 2.1 A multichannel neural recording system (Polystim) [34]

2.2 Neural Recording Devices

A typical implantable multichannel neural recording BMI contains three key parts: the front end, the signal processor and the signal transmission circuits. Figure 2.2 shows a typical neural recording BMI [35]. In this system, it can be seen clearly that this BMI contains a mixed-signal data acquisition part (front-end part), a spike detector (signal processing part) and the serial bus interface (signal transmission part). Signals are amplified and sampled in the mixed-signal data acquisition part. Then the system detects the neural spikes in the digital part. Finally, the

digitized data points are transmitted through the serial bus interface.

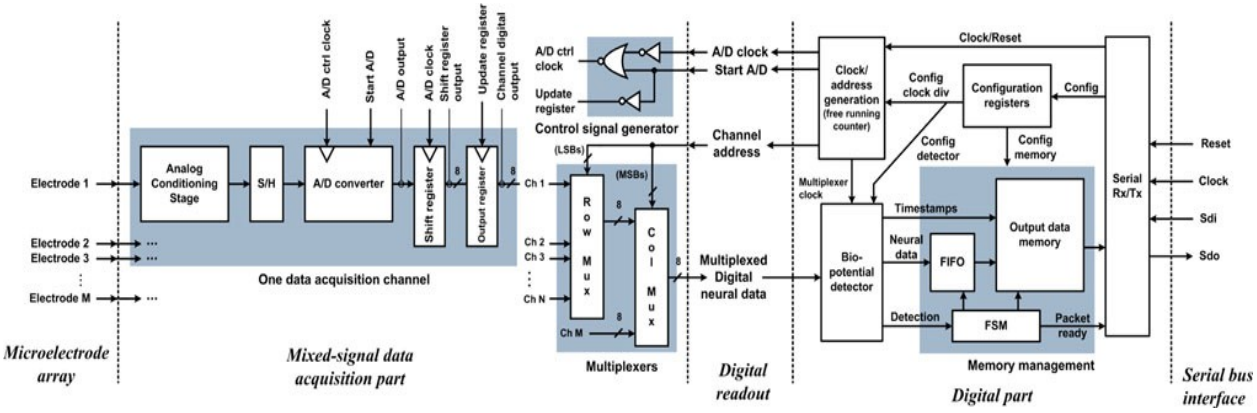


Figure 2.2 A typical neural recording BMI [35]

Generally speaking, the front-end circuit contains two parts: a signal preprocessor and an analog-to-digital converter (ADC). The signal preprocessor includes a signal amplifier and a filter. For example, Figure 2.3 shows a high-pass amplifier that provides a fixed gain of 50 dB and cut-off for all EEG signals with frequencies lower than 0.1 Hz, and power consumption of 99 μW [36]. Figure 2.4 shows a continuous-time OTA-capacitor (OTA-C) filter featuring 9th-order equiripple transfer functions with a constant group delay; the power consumption of this filter is only 360 nW [37]. Recent research, [38] introduced an 800 nW 43 nV/pHz neural recording amplifier using 0.18 μm CMOS technology with an area of 0.05 mm^2. Many recent articles concern the design of low-power high-performance amplifiers and filters [39] [40] [41].

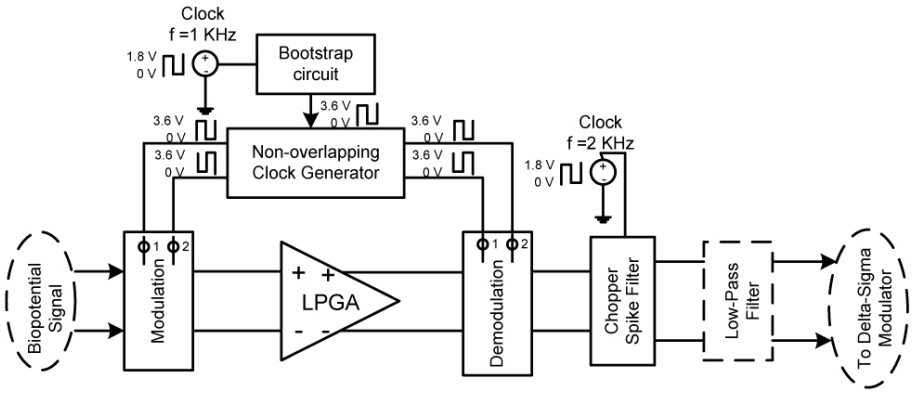


Figure 2.3 The chopped logarithmic programmable gain amplifier [36]

An ADC is needed for digitalized calculation and transmission. Figure 2.5 shows a nonlinear signal-specific ADC. Its sampling rate is 25 kS/s and its power consumption is only 87.2 μW

[42]. Other recent research outputs, [43] [44] contain details of the design of a high-performance ADC.

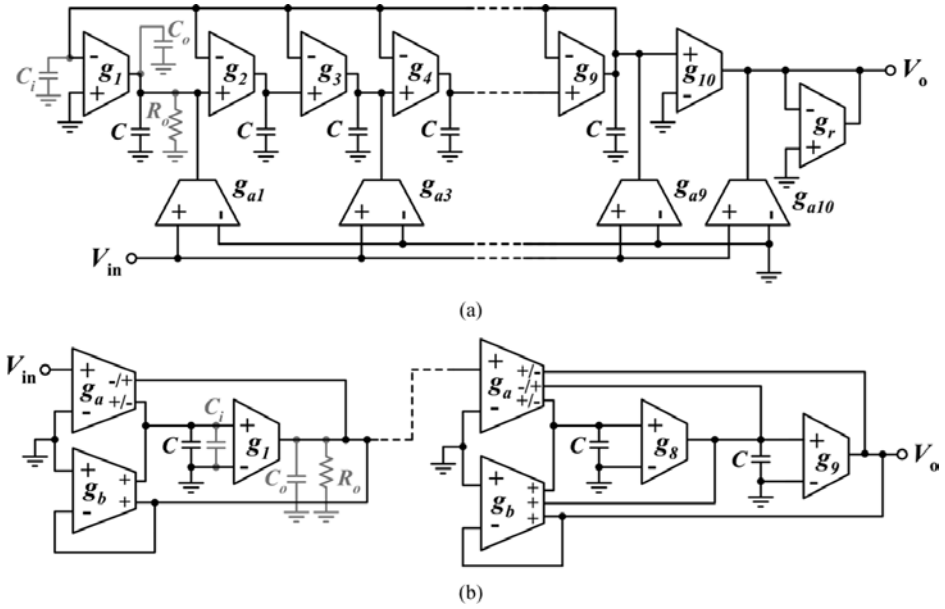


Figure 2.4 OTA-C continuous-time delay-filters, (a) the IFLF filter, (b) the cascaded filter [37]

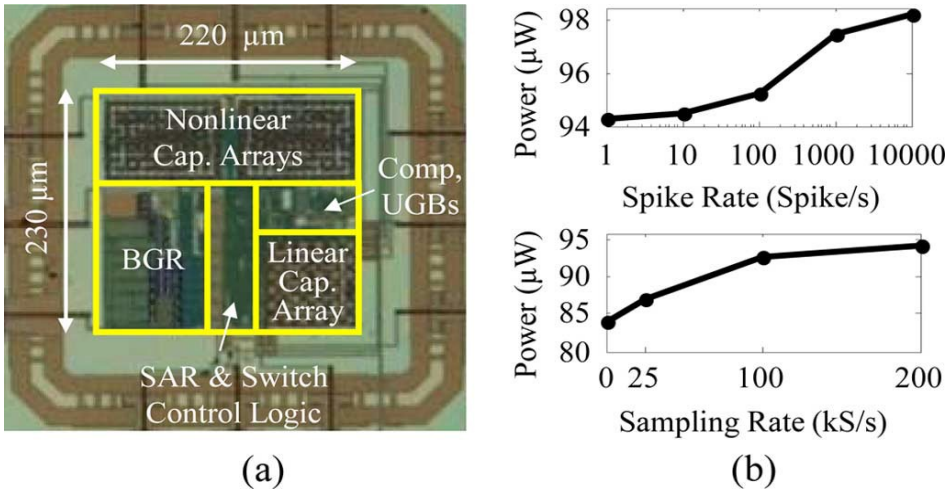


Figure 2.5 (a) Chip photograph of the proposed nonlinear ADC, (b) power consumption of the chip versus sampling rate (bottom) and spike rate at 200 kS/s (top) [42]

The second necessary component of an implantable neural recording BMI is the transceiver, and example of which is shown in Figure 2.6 [45]. This transmitter has a 1 GHz frequency band and a 20 Mb/s transmission rate. The power consumption of this transmitter is only 4.8 µW with a

0.9 V power supply. More information about the design of transceivers can be found from references [46] [47] [48].

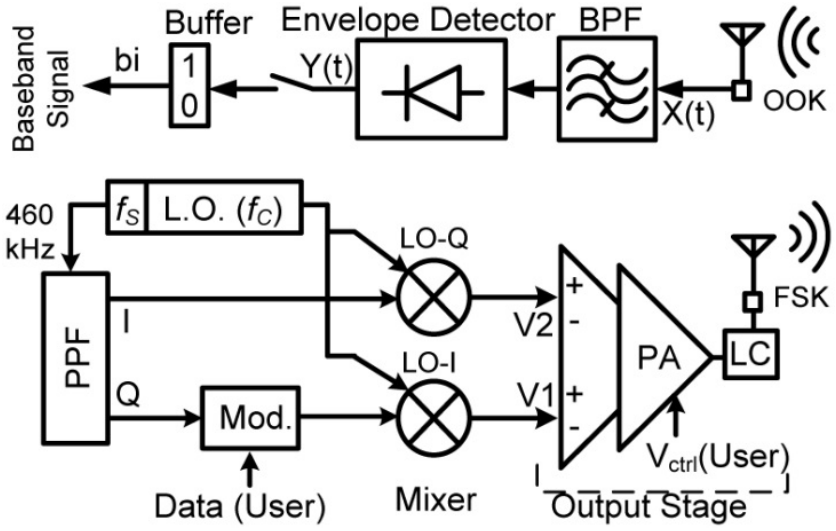


Figure 2.6 Block diagram of the proposed combined transceivers [45]

Thirdly, in a neural recording BMI, the signal processor is a very important part. It enables spike detection [29], feature extraction [49], and data compression [10] [50]. Spikes in brain activity can be used to study epilepsy [21] [51] ; in a typical epilepsy system, spikes detection is the first step in feature extraction [52]. In addition, spike detection can be used to study the activity of the neurons of the prefrontal cortex [53] [54] [55] [56]. Currently, most existing neural recording or stimulation systems integrate a spike detection function [57].

Although the literature on this topic is large, we focused on a comparison of the most recently published neural recording systems, which are not merely front-end but include detection, compression and transceiver circuits, or all of these. We describe five typical state-of-the-art neural recording systems in Table 2.1, and compare other systems outlined in articles published from 2007 to 2015 in Table 2.2. One system introduced in reference [58] is based on the analog-spike detector, and reference [59] introduces a system that uses only digital methods for signal processing. Two neural recording systems in references [54] and [60] use analog and digital methods to reduce or compress signals. The system described in [47] does not include signal processing.

Table 2.1 Performance comparison of five typical state-of-the art neural recording systems

Reference	[47]	[54]	[58]	[59]	[60]
Electrodes	10	1	100	128	64
Amplifier	Folded cascode OTA	Low-noise programmable gain OTA	Two-stage OTAs	OTAs	Two-stage operational amplifier
Gain (dB)	40	Adjustable between 47.5 and 65.5	60	60	60
Low and high cut-off frequency (Hz)	300, 8.13k	167, 6.9k	300, 5k	0.1-200 (low-frequency roll-off), 2k-20k (high-frequency roll-off)	<10–100, 9.1k
ADC	None	8-bit SAR ADC	10-bit SAR ADC	Adjustable 6-bit or 9-bit SAR ADC	8-bit ADC
Sampling rate (ksample/s)	None	90	15	40	7.8
Signal reduction	None	Analog detector and feature extraction	Analog programmable threshold	Nonlinear energy operator and feature extraction	Analog spike detector
Signal compression	None	None	None	None	Digital data compressor
Transmission	TDM-FM 433M Hz	None	FSK transmitter 433M Hz	Ultra wideband transmitter	FSK transmitter 4M/8M Hz
Size (mm)	22 × 11 × 5	0.4 ×0.4	4.7 × 5.9	8.8 × 7.2	14 × 15.5
Power consumption of the system (mW)	5	3.1×10^{-3}	13.5	6	14.4
Process Technology (μm)	0.5	0.18	0.5	0.35	0.5

Through comparing these neural recording systems, some conclusions can be drawn. First, signal processing systems are important. In the past, most implemented neural recording systems focused on the pre-processing, including the amplifier, filter and analog-to-digital data conversion. Some systems do not include any signal reduction and compression components or may integrate a simple spike detection system. Recently, lots of systems have begun to integrate more complicated signal processing systems, such as spike-sorting systems and CS-based signal compression systems; as noted previously, signal processing systems for neural recording devices have become a hot research topic.

Table 2.1 and Table 2.2 also show that designers are using more electrodes or channels for neural recording systems. Currently, to the best of our knowledge, the maximum number of electrodes used in a neural recording system without wireless telemetry and signals compression is 11,011 [61], which shows that huge numbers of electrodes can be used for a neural recording system, but with the limitations of power and size, the recorded data cannot be transmitted through wirelessly. Therefore, advances in neural signal processing are necessary.

Both analog and digital methods are used for signal processing. The digital method offers higher accuracy than the analog one. For example, for spike detection, the digital method can optimally implement the corresponding detection methods from the mathematical formulas to calculate thresholds, which means that the digital method is “smarter”, more flexible and has higher estimation accuracy [62].

Finally, as previously noted, a neural recording device must have low power consumption and small silicon area. The transmitter of a wireless multichannel implantable neural recording device consumes more energy than a wired device. There are two reasons for this problem. Firstly, huge amounts of data means that the system must use a high carrier frequency for transmission. Secondly, free carrier frequencies, known as ISM bands, are used to transmit the data, which increases the complexity of the transmission system. Conflicts between transmission and circuit performance can only be resolved by designing a low power and small area signal processing system.

Table 2.2 Comparison among different neural recording systems

Reference	[63]	[64]	[65]	[66]	[67]
Year	2015	2015	2015	2014	2014
Electrode	12	4	4	32	8
Amplifier	OTA	Two-stage amplifier	Two-stage amplifier	Intan Technologies RHA2132 amplifier chip	Operational amplifier
Gain (dB)	40, configurable	Configurable 230 − 6 k	72	200	55.7 / 50.3(AP), 50.3 / 45.1(LFP)
Low and high cut-off frequency (Hz)	Configurable	None	30k (H)	0.17, 4.5k	0.12 – 3k, 20 – 2k
ADC	SAR ADC (12 bits)	SAR ADC (10 bits)	SAR ADC (8 bits)	AD7980	None
Sampling rate(ksample/s)	10^3	20	100	31.25	None
Signal reduction	None	None	Energy-based	MAD, template matching	Analog spike detector
Signal compression	CS	CS	None	None	None
Transmission	None	None	FM/FSK	None	None
Power consumption (μW)	4.6g lithium battery,70 hours	16 per electrode	8000	None	4.8 per channel
Size (mm)	4.5×1.5	0.11 mm^2 per electrode	1.5×0.75	29.5 × 43.3	1.5 × 1.5
Process technology (μm)	0.18	0.18	0.5	None	0.18

Table 2.2 Comparison among different neural recording systems (cont'd)

Reference	[68]	[41] [69]	[70]	[71]	[72]
Year	2014	2013	2013	2013	2012
Electrode	4	100	64	1	10 × 10
Amplifier	Fully-differential amplifiers	OTA	OTA	OTA	Capacitive-feedback, folded cascode OTA
Gain (dB)	43 – 80	34 – 40	54 – 60	39.6	46
Low and high cut-off frequency (Hz)	0.1, 2000	432, 5.1k	None	0.8, 5.2k	0.1, 7.8k
ADC	SAR ADC (8 bits)	SAR ADC (9 bits)	SAR ADC (8 bits)	Sigma-delta ADC (13 bits)	SAR ADCs (12 bits)
Sampling rate (ksample/s)	10 – 100	24.5 – 245	20	2000	20
Signal reduction	None	Analog spike detection	Feature extraction	None	None
Signal compression	None	None	None	None	None
Transmission	MICS/ISM-compliant transmitter digital FSK	Burst-mode wideband FSK	All-digital pulsed ultra wideband	Standard series peripheral interface	FSK with 3.2/3.8GHz wireless transmitter
Power consumption (μW)	1100	1160	16 per electrode	2760	90.6
Size (mm)	8.6 × 9.7	4.5×1.5	4 × 3	11.25 mm^2	5.2 × 4.9 (preamplifier) 2 × 2 (controller)
Process technology (μm)	0.13	0.18	0.13	0.6	0.5

Table 2.2 Comparison among different neural recording systems (cont'd)

Reference	[73]	[74]	[75]	[76]	[77]
Year	2012	2012	2011	2011	2011
Electrode	32	14	1	1	16
Amplifier	Two-stage, band-pass, low-noise amplifier	Low-noise, low-power amplifiers	OTA	Instrumentation amplifier	Commercial acquisition system
Gain (dB)	66.5	500 V/V	100	300,500, 900,1300	76
Low and high cut-off frequency (Hz)	Adjustable, 9.6k (H)	250, 10k	300, 5.2K	None	300 (L)
ADC	ADS7953 from Texas Instrument (12 bits)	SAR ADCs (11 bits)	SAR ADC (9 bits)	SAR ADC (11 bits)	SAR ADC (8 bits)
Sampling rate (ksample/s)	Maximum 62.5	26.1	11.52	64 or 1024	None
Signal reduction	Nonlinear energy operator	None	None	Feature extraction	Setting threshold
Signal compression	None	None	None	Adaptive sampling	None
Transmission	None	RF transmitter with 902-928 MHz carrier frequency	905 MHz FSK transmitter	None	Manchester coded frequency shit keying 400M carrier frequency
Power consumption (μW)	None	1230	1.5v silver-oxide batteries, runs 5 hours	30 for ASP	17200
Size (mm)	None	2.36 × 1.88 × 0.25	6 × 5	4.6 × 4.5	None
Process technology (μm)	None	0.35	0.6 and 0.35	0.5	0.5

Table 2.2 Comparison among different neural recording systems (cont'd)

Reference	[78]	[79]	[80]	[57]	[81]
Year	2010	2010	2010	2009	2009
Electrode	64	1	18	10 × 10	16 × 16
Amplifier	Low-noise, band-pass pre-amplifier	Two-stage low noise OTA	Three-stage instrumentation amplifier	User-selectable amplifier	OTA
Gain (dB)	65 – 83	50 – 80	72	60	48 – 68
Low and high cut-off frequency (Hz)	10, 10k	0.1 – 1k, 8k, adjustable	< 1, 200	250, 5k	0.01 – 70, 500 - 5K
ADC	SAR ADC (8 bits)	Commercial component TI MSP430	SAR ADC (12 bits) with power-gating	SAR ADC (10 bits)	Sample-and-hold circuit (8 bits)
Sampling rate (ksample/s)	20	20	0.6, maximum is 100	15.7	< 10k
Signal reduction	Adaptive threshold	Adaptive absolute threshold	Spectral energy distribution extraction	Adaptive threshold	None
Signal compression	None	None	None	None	Delta compression lower the resolution
Transmission	Narrowband 400-MHz binary Manchester-coded FSK	Power carrier frequency is 13.56MHz. data carrier frequency is 433/915MHz	None	FSK modulation with carrier frequency 902-928 Hz	None
Power consumption (μW)	16600	< 4850	9 uJ/ per feature-vector	8000	5040
Size (mm)	3.1 × 2.7	4.9 × 3.3	2.5 x 2.5	5.4 × 4.7	3.5 × 4.5
Process technology (μm)	0.35	0.5	0.18	0.6	0.35

Table 2.2 Comparison among different neural recording systems (cont'd)

Reference	[82]	[83]	[84]	[85]	[86]
Year	2009	2008	2008	2007	2007
Electrode	16	32	16	64	16 × 16
Amplifier	OTA	Low noise OTA	Two-stage voltage amplifier	Low noise multiplexed amplifier array	OTA
Gain (dB)	45.7, 49.3, 53.7 or 60.5	48	39.6	64	48 – 68
Low and high cut-off (Hz)	1, 12K	1, 7k	0.2 – 94, 140 – 8.2k	6 – 1k, 3 – 15k	0.01 – 70, 500 – 5k
ADC	SAR ADC (10 bits)	Normal ADC (5 bits, 10 bits)	Gm-C incremental ΔΣ ADC (8-12 bits)	AD7277 (10 bits)	Off-chip ADC
Sampling rate (ksample/s)	256	22	< 16	40	10
Signal reduction	None	Analog spikes detection	Analog filtering	Digital filters based detection	Wavelet decompos-ition
Signal compression	None	None	None	None	None
Transmission	None	Bluetooth	Wireless power harvesting and telemetry system	USB 2.0	None
Power consumption (µW)	60.3	109.58	1800	33 000	5.04 (with-out wavelet processor)
Size (mm)	2.5 × 3.3	3 × 3	3 × 3	2.8 × 3.2 (amplifier array) 40 × 60 (FPGA)	4.5 × 3.5 (only interface)
Process technology (µm)	0.35	0.6	0.5	0.35	0.35

2.3 Neural Signal Processing

Signal processing techniques are used for analysis or perception of physiological measurements. The purpose of acquiring physiological signals is to gain insight into the system that produces these signals. As noted above, these signals may be acquired in electrical form. A typical schematic block diagram of physiological measurement is shown in Figure 2.7. Most of the in situ signal processing follows the process in this diagram. First of all, the designer needs to know the physiological process and design the corresponding signal collector. This collector can be the electrodes, the sensor or the chemical indicator. After collection of the signals, these signals need to be translated into electrical signals. Then, an analog preprocessing can amplify the signal or remove noise. After the analog preprocessing, a digitized conversion prepares for the following calculation, transmission or analysis. If a designer uses analog signal processing in the system, then an analog processor will be added before the converter; if the digital signal processing is applied for the system, then a digital processor is integrated after the converter. Finally, the digitized signals are analyzed in a computer.

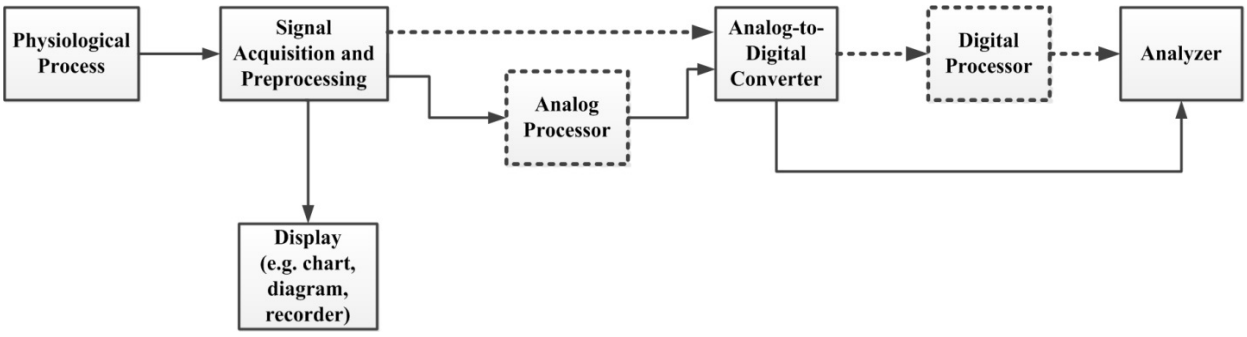


Figure 2.7 Diagram of physiological measurement

For a multichannel neural recording device, handling the quantity of the recorded data is one of the most difficult problems that must be solved. Designers need to considerably reduce large amounts of data, without degrading the data quality, for easy transfer through wireless transmission. To solve this problem, two strategies can be applied: signal reduction and compression. Signal reduction methods retain most of the information of the signal but remove useless signals; for example, spike detection is a signal reduction method. Signal (or Data) compression methods use one of many possible approaches to acquire a subset of signals, then based on this subset, apply an algorithm to recover the original signals. In the following two

sections, two processing strategies, spike detection (and sorting) and signal compression, are reviewed.

2.3.1 Spike Detection and Sorting

Neural signals, produced by the neurons in the brain, can be recorded as bioelectrical signals. Bioelectrical signals are one kind of important biomedical signals that reveal the behavior of relevant organs. The basic component of all bioelectrical signals is action potential (AP) [87]. AP is caused by the flow of sodium (Na+), potassium (K+), chloride (Cl-) and other ions moving across every cell membrane [13]. The AP provides information on the nature of physiological activity at the cell level. When a single neuron is stimulated by a neural or external electrical current, it produces APs. Recording an AP requires the isolation of a single neuron, then microelectrodes with tiny tips are used to stimulate the neuron or record the response [87].

Extracellularly recorded neural signals have some important characteristics. Firstly, extracellularly recorded APs are called spikes [88]. Neurons communicate with each other using spikes. Each spike has an amplitude of about 70 mV (relative to the extracellular fluid) and a duration of around 1 – 2 ms [89] [90]. When an extracellular microelectrode is held tens of microns away from the neurons, a value of 50 – 500 μV can be detected. A typical neuron generates 10 – 100 spikes per second [29] [58]. (Figure 2.8 shows APs collected from multiple neurons of an adult male rhesus macaque monkey). In addition, once a neuron generates an AP, there is a period during which a cell cannot respond to any new stimulation, known as the absolute refractory period (about 1 ms). This is followed by a relative refractory period (several microseconds), and in this period, another AP may be triggered by a stronger stimulation [13].

Spike detection is an important step in many processes and analyses involved in investigating the activity of the nervous system. First, the spike detection process detects APs (spikes) that are immersed in background noise during extracellular recordings of neural signal. This process enables interpretation of neuronal activity and decoding of the included information. Furthermore, spike detection is a very useful reduction process for transmission in a wireless multichannel implantable neural recording system.

Spike detection processing involves two important concepts. The first one is concept of online and offline detection. Online detection means that the neural spikes are detected at the same time

as the signals are recorded. Offline detection means that detection occurs after recording [88]. This method usually has a long delay, and it is obviously not suitable for a multichannel implantable neural recording system. The second important concept is adaptive detection and the detection of manual setting. The detection of manual setting means that the threshold or template is not calculated from the previously recorded data but is directly set by the designer. In the early days of spike sorting, spike detection was done purely in analog hardware. It was typically performed using a simple voltage trigger, with a voltage threshold set manually by the user [88]. If the voltage of the signal crossed that threshold, a pulse would be generated to indicate the presence of a spike [91]. This method is appealing because of its simplicity, and, as a result, is still used today by many research groups, especially with analog designs [88]. The largest disadvantage of the manual setting method is its requirement for thresholds or templates in advance. In contrast, adaptive detection means that the threshold or template that is used for the detection is determined from the previously recorded data. Comparing with the manual setting method, this method not only runs automatically to detect the spikes but allows estimation of the threshold, which is definitely much more advanced [92]. Currently, neural recording systems usually integrate an online adaptive spike detection component [93] [94] [95]. Furthermore, if the threshold can be acquired in a short time, this method can be called real-time detection [90]. Automatic calculations of the threshold usually involve a training period, and they still can be considered to be real-time processing.

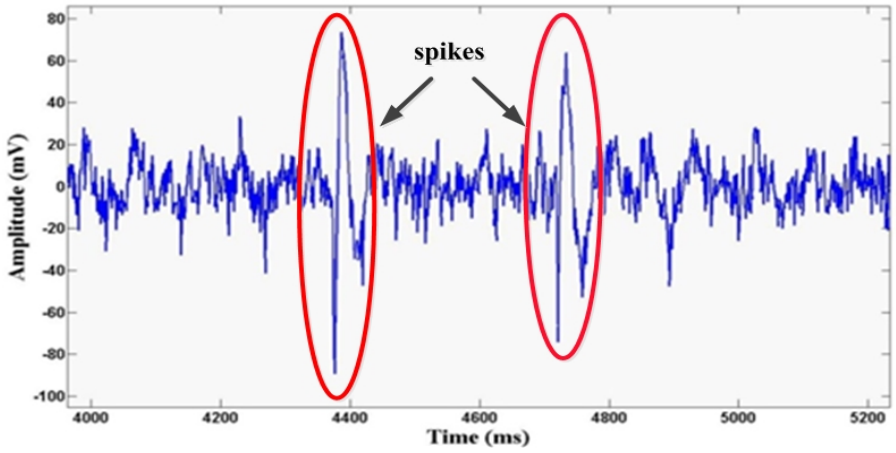


Figure 2.8 Recorded signals from an adult male rhesus macaque monkey

Spike detection techniques can be divided into amplitude-based, energy-based and template matching methods. These methods are described in the following sections.

2.3.1.1 Amplitude-based Spike Detection

Detecting neural spikes from background noise is commonly done by comparing signals' amplitudes with thresholds. This method was initially used to analyze offline neural signals [91]. now, amplitude-threshold-based detection is a commonly used online adaptive detection method. The idea of this method is that a spike is a sudden pulse and its amplitude is obviously larger than the ambient signals. The noise of the signal is usually regarded as the Gaussian white noise, and the threshold is the standard deviation multiplied by a coefficient [96]. Based on this idea, several real-time, adaptive spike detection methods are emerging, such as RMS methods [97], median absolute deviation (MAD) [98], EC-PC [99], cap fitting [100], and maximum and minimum spread (MMS) estimation [96] methods. The RMS estimator is traditionally used to estimate the standard deviation of the background noise, which is shown in Figure 2.9(a) [97]. The threshold is calculated using (2.1).

$$T = P \times \sqrt{\frac{1}{N}\sum_{1}^{N}(x_n - \bar{x})^2} \qquad n = 1\ldots\ldots N \tag{2.1}$$

where $\bar{x}$ is the mean value.

The second method is called MAD estimator, which is shown in Figure 2.9(b) [98]. The threshold is calculated in (2.2).

$$T = P \times \frac{median(|x_n - \bar{x}|)}{0.6745} \qquad n = 1\ldots\ldots N \tag{2.2}$$

All the samples in a given time frame are subtracted by their mean value. Then the absolute values of the subtracted samples are sorted. The MAD is defined as the median value of $|x_n - \bar{x}|$. The standard deviation of background noises is equivalent to the MAD divided by 0.6745.

The next estimator, shown in Figure 2.9(c), is the cap fitting estimator [77]. The threshold T is set as $P \times \sigma$, where σ is the standard deviation. If T exceeds $X_{0.9987}$, the distribution of data below T is considered to be Gaussian noise. Otherwise, the samples that exceed T are regarded as neural spikes and are removed from the signal.

The final estimator, shown in Figure 2.9(d), is the MMS sorting estimator. This estimator has been proven to have better performance than the other three estimators [96]. This method firstly finds the maximum and minimum value of the data during a given time frame (called a window). Then the difference between maximum and minimum value is calculated and stored in a buffer.

When the buffer is full, all of the data are sorted, and a subset of the sorted data is averaged. Finally, the averaged value is multiplied by an amplification coefficient.

The threshold estimator can be designed as an analog or a digital circuit. An analog self-time static spike detector is shown in Figure 2.10 [101], and a digital spike detector, called an mMMS sorting estimator, is shown in Figure 2.11 [102].

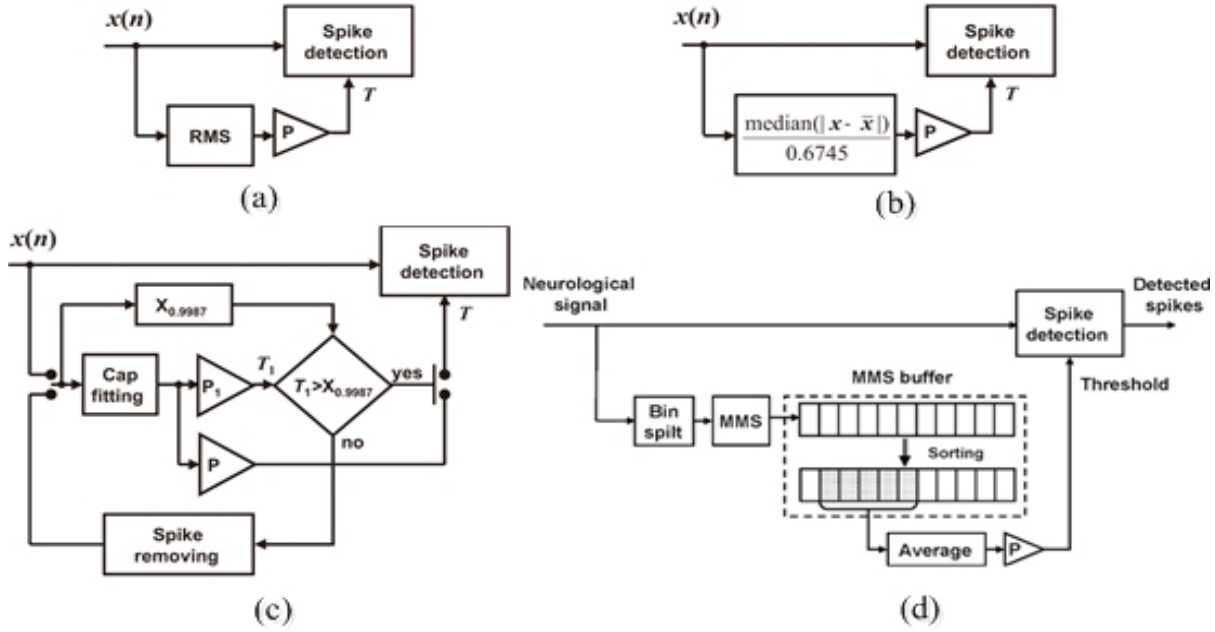


Figure 2.9 Comparison of four digital estimators, (a) RMS estimator, (b) MAD estimator, (c) Cap fitting estimator, (d) MMS estimator [96]

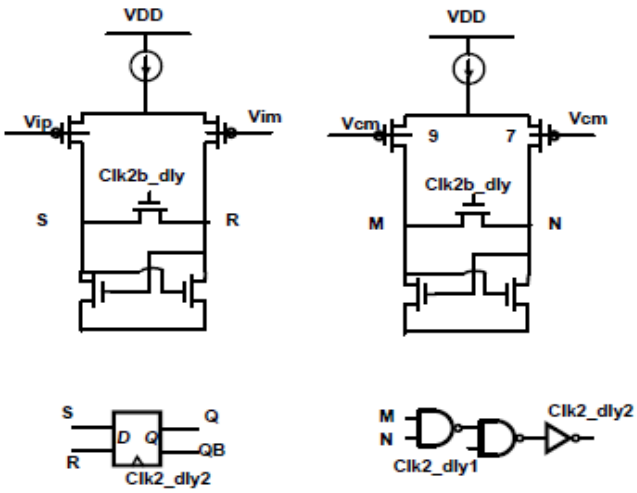


Figure 2.10 An analog self-timing static detector [101]

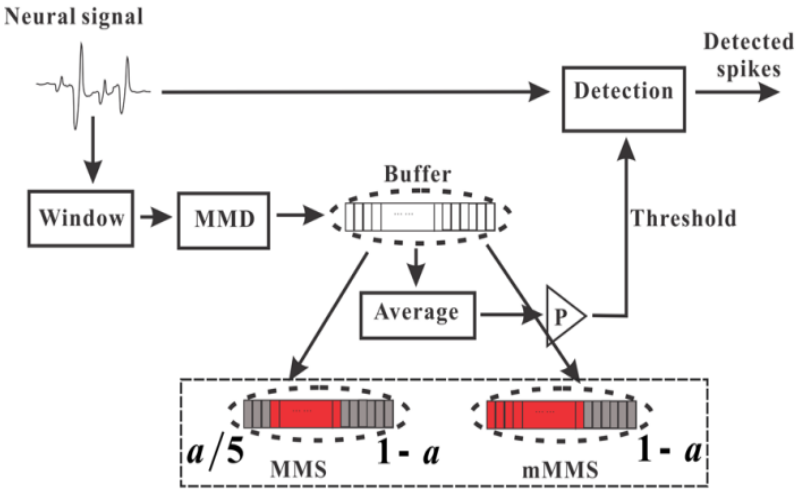


Figure 2.11 Digital mMMS estimator [102]

2.3.1.2 Energy-based Spike Detection

The energy-threshold-based spike detection method is based on the idea that when a spike happens, the energy of the signal has a sudden change, so the system can calculate the threshold of the change in its energy. An important form of the energy-threshold-based spike detection method is called teager energy operator (TEO) or nonlinear energy operator (NEO) [103]. When the signal is processed in a time-domain or frequency-domain window based on the TEO, this method is called the smooth TEO (STEO), which has better estimation accuracy than the TEO [103]. Equations for the TEO and STEO are established by (2.3) and (2.4).

$$\psi[x(n)] = x^2(n) - x(n+1)x(n-1) \tag{2.3}$$

$$\psi_s[x(n)] = \psi[x(n)] \otimes w(n) \tag{2.4}$$

where $\otimes$ represents the convolution operator and $w(n)$ represents the window.

The threshold, given in reference [103], is shown in (2.5).

$$T = C\frac{1}{N}\sum_{n=1}^{N}\psi_s\,[x(n)] \tag{2.5}$$

where N is the number of samples and C is the scaling factor.

The authors in [90] describe another method to calculate the threshold, which is shown in (2.6) – (2.8). In this article, they chose the Hamming window $w_H(n)$, having the following value $w_H(n) = [0.08, 0.54, 1, 0.54, 0.08]$.

$$T_{\psi_s} = \mu_{\psi_s} + p\sigma_{\psi_s} \tag{2.6}$$

$$\mu_{\psi_s} = 2.24(r_{xx}(0) - r_{xx}(2)) \tag{2.7}$$

$$\sigma^2_{\psi_s} \approx 4.8r^2_{xx}(0) + 0.7r^2_{xx}(1) + 4.4\,\mathrm{r}^2_{xx}(2) + 0.6\,\mathrm{r}^2_{xx}(3) - 9.3\,\mathrm{r}_{xx}(0)\,\mathrm{r}_{xx}(2) - 1.2\,\mathrm{r}_{xx}(1)4.8\,\mathrm{r}_{xx}(3) \tag{2.8}$$

where $r_{xx}(m)$ is the autocorrelation of $x(n)$ at lag m.

Some system or circuit designs are based on the TEO or STEO method. For example, the authors in reference [90] propose a spike detection system based on the STEO method, which is shown in Figure 2.12. In reference [104], the authors give the implementation of the STEO method without threshold estimation. The power consumption of the relevant spike detection module in this article is reported as around 1 μW. Also, some designers use the TEO method to design digital (Figure 2.13) [105] or analog spike detector devices (Figure 2.14) [106].

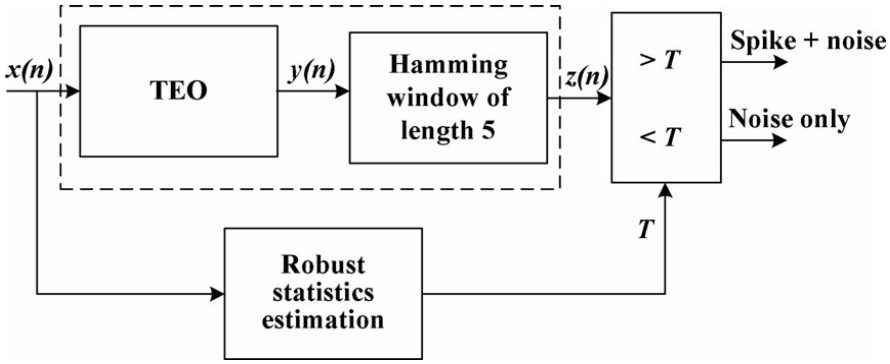


Figure 2.12 Block diagram of STEO-based spike detection with adaptive threshold [90]

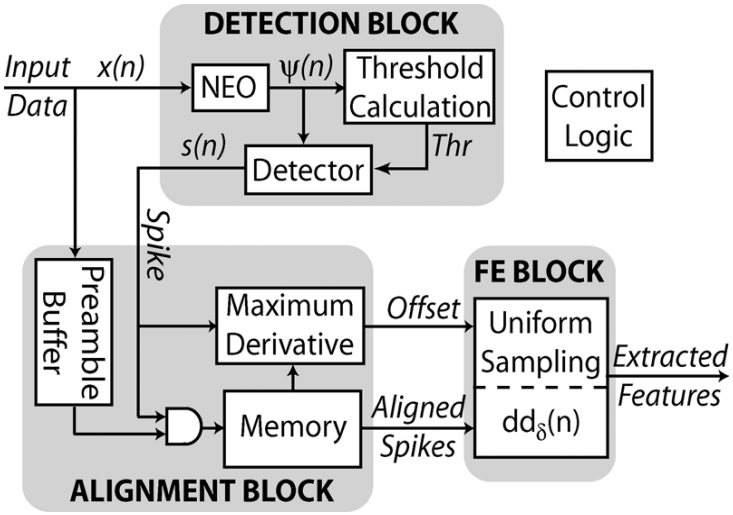


Figure 2.13 Diagram of neural spikes sorting system using TEO spikes detection method [105]

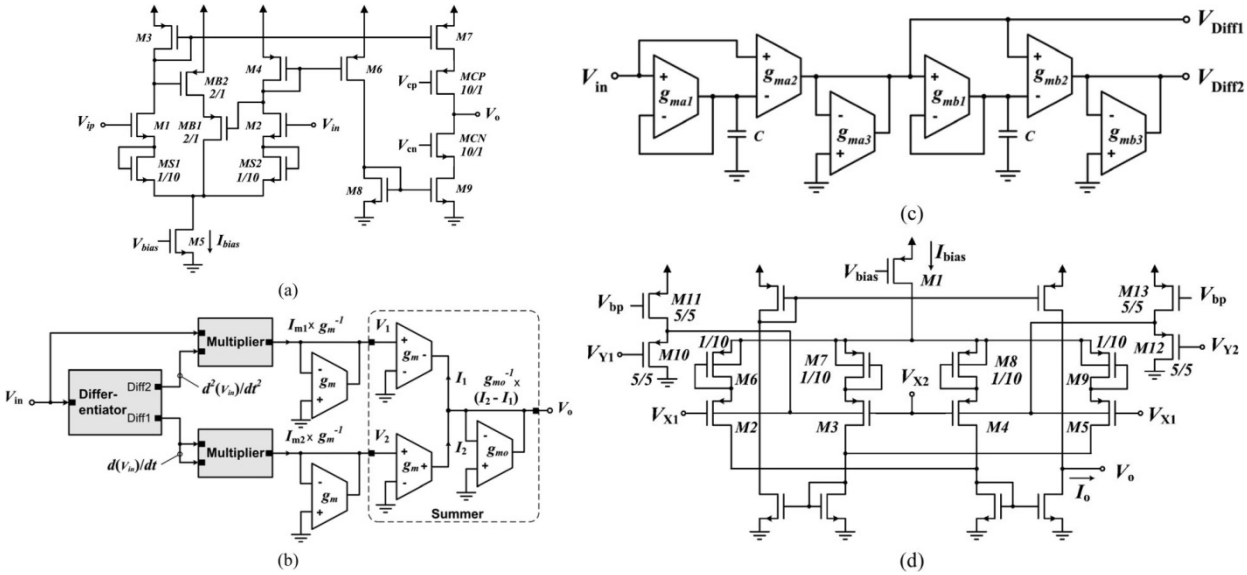


Figure 2.14 Building blocks synthesizing the TEO-based preprocessor, (a) subthreshold OTA with source degeneration and bump linearization devices, (b) top-level diagram of the TEO preprocessor, (c) the differentiator circuit, (d) four-quadrant analog multiplier [106]

2.3.1.3 Spikes Detection with Template Matching

Template matching finds segments of the signals that are similar to the given spike templates. The template matching method is usually complicated, as it contains a convolution and Fourier transformations. This method needs a priori knowledge of spike templates and requires the user to specify a threshold for similarity. In earlier times, some researchers used Euclidean distance [107] and cross-correlation [108] to detect spikes with known templates. A typical template matching technique is matching filter [109] [110]. The discriminant of the matching filter is shown in (2.9):

$$D_1 = x_n^T T \ \geq \ \lambda \tag{2.9}$$

where x_n^T is one segment of the signal, T is the template and λ is the threshold. Reference [111] uses likelihood ratio detection (LRT) to make the template matching-based detection. The discriminant is shown in (2.10) and (2.11)

$$D_2 = x_n^T \Sigma^{-1} T \underset{\mathrm{H_0}}{\overset{\mathrm{H_1}}{\gtrless}} log\frac{P(H_0)(c_{10}-c_{00})}{P(H_1)(c_{01}-c_{11})} \equiv \lambda \tag{2.10}$$

$$\begin{aligned} \mathrm{H}_0: &\quad x_m = n \\ \mathrm{H}_1: &\quad x_m = T + n \end{aligned} \tag{2.11}$$

where n is the white Gaussian noise, T is the template, c_{ij} is the cost of deciding hypothesis. In reference [112], the authors introduce a normalized correlator to make the detection, which is shown in (2.12) — (2.14).

$$D_3 = \bar{x}_n^T \bar{T} > \lambda \tag{2.12}$$

$$\bar{x}_n = x_n / \|x_n\| \tag{2.13}$$

$$\bar{T} = T / \|T\| \tag{2.14}$$

where λ can be chosen as 0.5.

For all of these template matching methods, choosing the threshold is important. Reference [112] reports that the threshold can be chosen as $\sigma^2 \times P$, where σ is the standard deviation and P is a coefficient. Several recent articles discuss using Bayesian inference to determine the threshold [113] [114].

As the templates are rarely known in advance, the template matching spike detection system must automatically generate templates. In reference [115], the authors present a semiautomatic template matching spike detection system, in which the designer must manually determine the final number of spike templates. Reference [112] introduces another spike detection method with automatic template matching; the final clustering step is implemented by a method called the Osort algorithm [116]. Figure 2.15(a) and Figure 2.15(b) show this spike detection method and the Osort algorithm. In reference [117], the authors put forward an template matching algorithm that is composed of four main components.

In conclusion, spike detection is a crucial component of a BMI; it not only provides the prerequisites for analysis of recorded signals, but reduces the quantity of recorded data. In reference [88], the authors compare all three methods of spike detection, maintaining that the detection accuracy of adaptive amplitude-based and energy-based spike detection methods are almost identical, but noting that the amplitude-based method has a simple structure. In reference [90] [104], the authors assert that the energy-based spike detection method is better than the absolute amplitude spike detection method. Our own research and some other work [116] [118]

have shown that template matching methods have better detection accuracy than amplitude-based or energy-based methods.

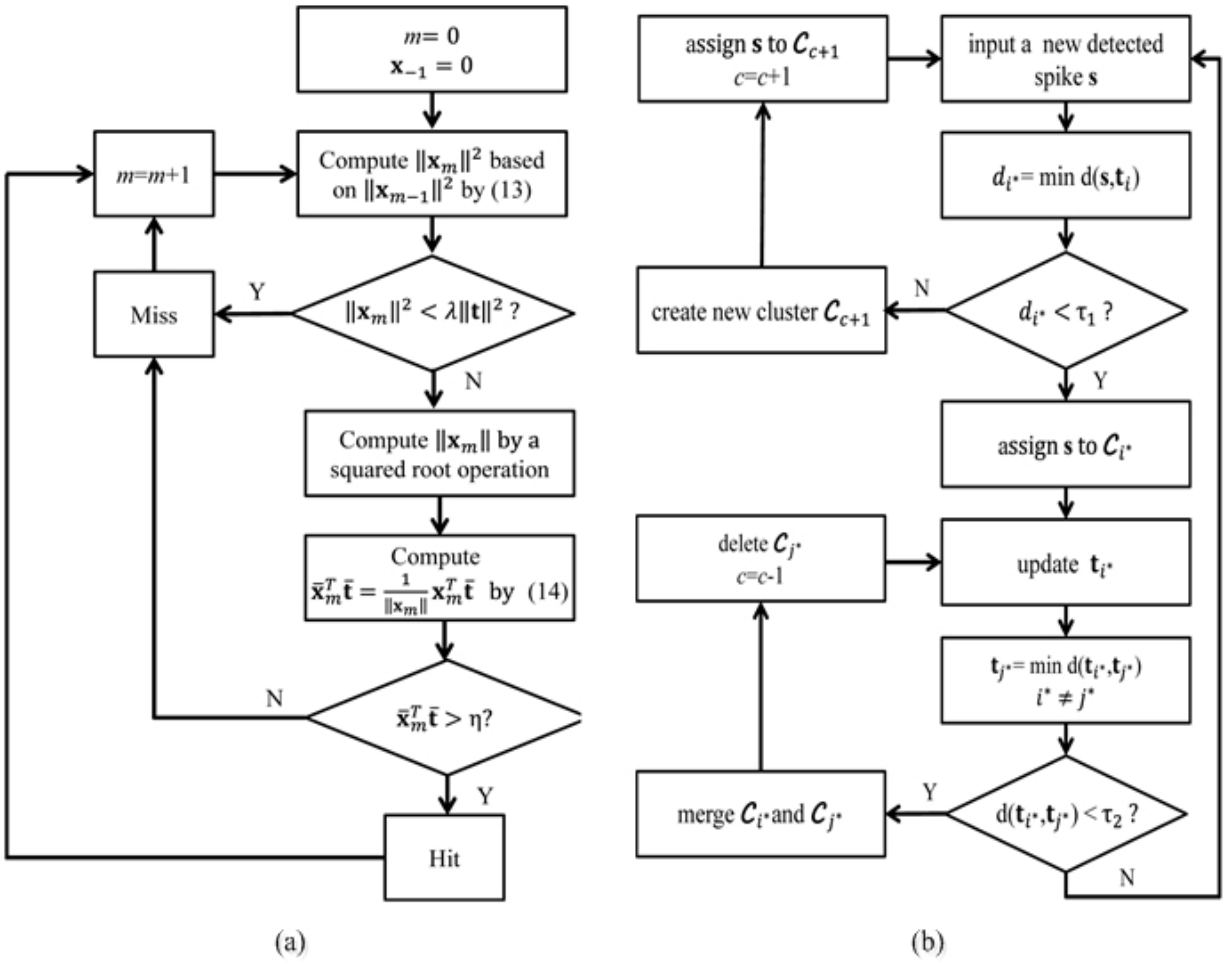


Figure 2.15 An automatic template matching spike detection method, (a) the proposed template matching spike detection method [112], (b) the Osort algorithm [116]

2.3.1.4 Spike Sorting

To study the activity of neurons, the researcher usually needs to understand single-unit activity to learn how a type of neuron responds to a specific stimulus. Some neural signal processing algorithms operate on signals from individual neurons [119] [120]. But because of the size of the recording electrodes, the recorded spikes are usually from several neurons; therefore, a spike-sorting method is needed. Figure 2.16 shows a diagram of the spike-sorting process [88]. This process has three main steps: feature extraction, dimensionality reduction and clustering.

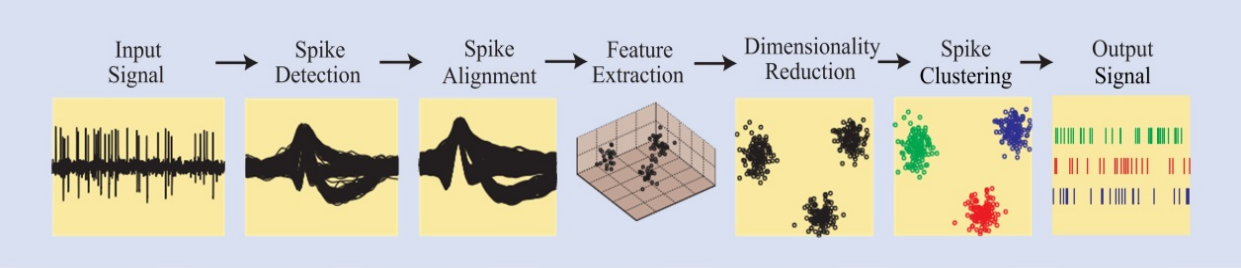


Figure 2.16 The spike sorting used to obtain single-unit activity [88]

Feature extraction is used to analyze the information within spikes, and it also acts as a signal reduction method in a neural recording device. There are many methods of feature extraction, such as the extraction of the maximum amplitude or width of the spike [91], principal components analysis (PCA) methods [121], first and second derivative extrema [122], and the discrete wavelet transform method [123].

Dimensionality reduction reduces the complexity of the clustering, leaving only the necessary features and increasing the accuracy of clustering. Several methods of dimensionality reduction exist, such as the Lilliefors test [124], and Hartigan's dip test [49].

The final step of the spike-sorting process is spike clustering. The clustering method puts spikes with similar features together. However, it is very difficult to judge similarity in online adaptive spike processing, so the unsupervised spike clustering method is very complicated. Various spike clustering methods exist, such as the K-means method [91] and valley seeking [125].

Finally, numerous different implementations of spike-sorting systems exist. Figure 2.17 shows a spike-sorting BMI; the authors used TEO-based spike detector and derivative-based feature extraction methods [59]. Figure 2.13 shows a 64-channel spike-sorting device; its power consumption is 2.03 μW/channel and its area is 0.06 mm^2/channel [105]. In [126], the authors present the hardware architecture of a spike-sorting device which uses PCA for feature extraction and the K-means method for spike clustering.

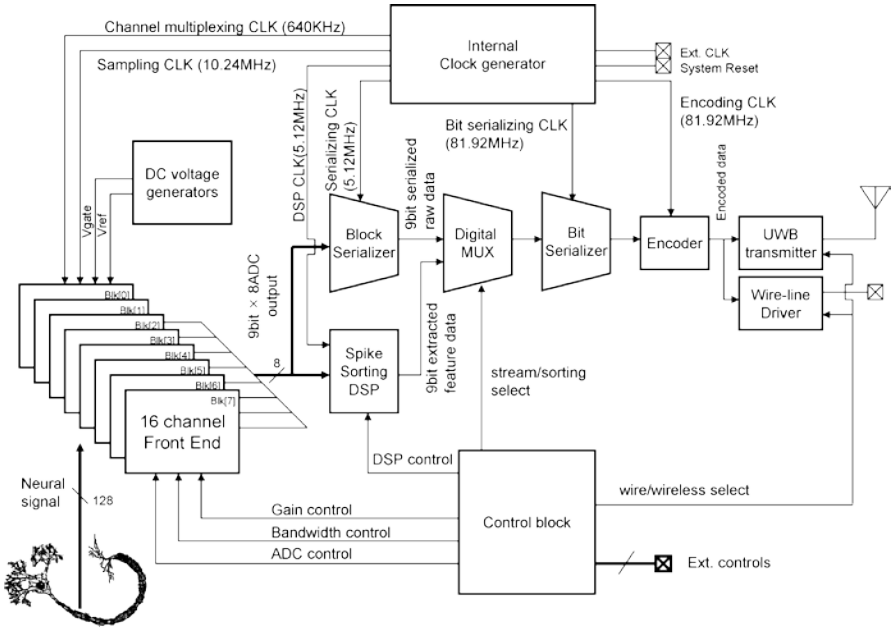


Figure 2.17 Block diagram of an integrated neural recording system with spike sorting [59]

2.3.2 Signal Compression with CS Technique

In the last section, we reviewed spike detection and sorting methods. Signal compression methods can recover original signals, so can keep more information about the recorded signals, and are obviously better in some situations which require original signals. For these reasons, signal compression has attracted considerable attention in the BMI design field [127] [128] [129].

Compressed sensing (CS) is a new signal compression technique which shows great potential for compressing neural signals. In recent years, CS has been a very hot topic in the areas of applied mathematics, computer science and electrical engineering [130] [131]. The CS concept was hintingly discussed more than one hundred years ago, but it has only recently gained scientific interest due to some theoretical breakthroughs [130]. In the 1900s, Carathéodory proposed a theory that includes the concept of reducing the amount of the sampled data [132]. More recently, Candès and Donoho have shown that a signal with a sparse representation can be recovered with fidelity [133].

When a signal is a sparse signal, there is no need to use the traditional Nyquist rate to sample the data; the CS technique can be used instead. CS includes sparse signals, sensing matrices and reconstruction methods. Figure 2.18 shows the CS technique [134]. The process of CS can be described as follows: signals are processed to find its sparse presentation (approximation or

coordinate conversion); then they are compressed by a random or deterministic sensing matrix [130] [135]; finally, the compressed signals are recovered through a reconstruction algorithm [136] [137]. CS is known to be effective for neural signal compression; previous authors have used CS to build BMIs. In [138], the authors show how to apply CS to design an analog or digital circuit, and use a digital circuit to fulfill the CS, which is shown in Figure 2.19. In [8] [11] [134], the authors put forward analog or digital circuits that apply the CS technique. Finally, some recent articles show how to use CS to compress non-sparse signals [139]. CS is discussed further in the following sections.

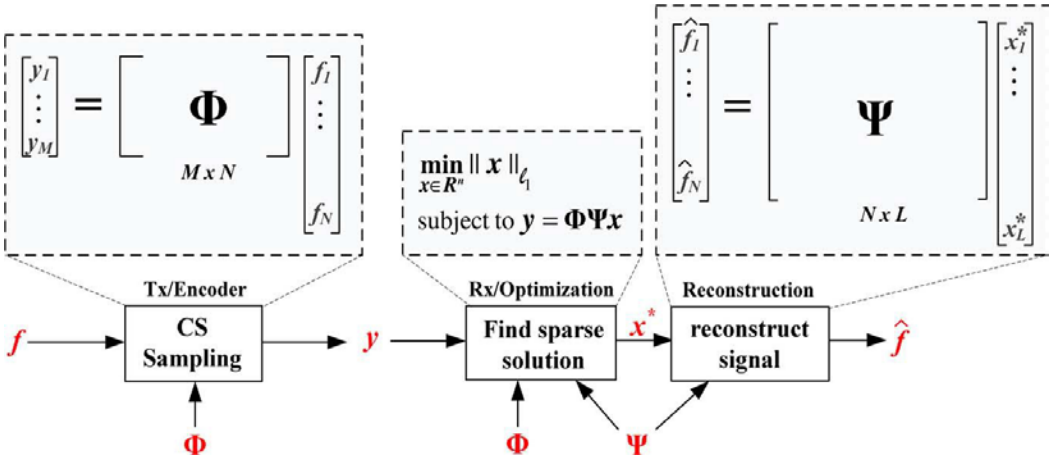


Figure 2.18 Diagram of CS sampling framework [134]

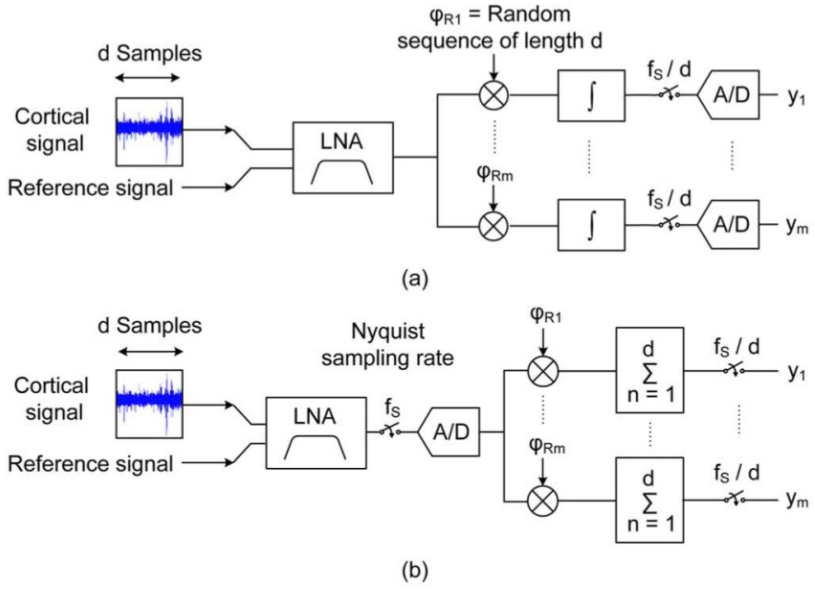


Figure 2.19 Block diagram of (a) the analog single-channel CS, (b) the digital single-channel CS [138]

2.3.2.1 Introduction to Compressed Sensing Theory

In this section, we will introduce the three most important concepts in CS: sparse signals, sensing matrix and signal recovery algorithms.

1. Sparsity and compressible signal

Mathematically, a signal *x* being *k*-sparse denotes that it has at most *k* non-zeros, which is expressed in (2.15) [140].

$$\Sigma_k = \left\{ x : \|x\|_0 = |\mathrm{supp}(x)| \le k \right\} \tag{2.15}$$

where $\mathrm{supp}(x) = \left\{ i : x_i \neq 0 \right\}$ denotes the support of *x* and $|\mathrm{supp}(x)|$ denotes the cardinality of $\mathrm{supp}(x)$. The CS technique does not deal directly with signals that are not obviously sparse, but admits a sparse representation in some basis Φ. Therefore, referring to *x* as being *k*-sparse denotes that $\|c\|_0 \le k$, when $x = \Phi c$, and c is the coefficient.

An important point in practice is that not all real-world signals are truly sparse; therefore, being compressible means that they can be approximately sparse or relatively sparse in various contexts. A way to measure a signal and their "compressible signal" x' ($x' \in \Sigma_k$) is expressed as (2.16) [130].

$$\sigma_k(x)_p = \min_{x' \in \Sigma_k} \|x - x'\|_p, \text{ if } x \in \Sigma_k, \text{ then } \sigma_k(x)_p = 0. \tag{2.16}$$

Another way to think about the compressible signals is to consider the rate of the decay of their coefficients. Specially, if $x = \Phi c$ and we sort coefficients c_i such that $|c_1| \ge |c_2| \ge \cdots \ge |c_n|$, then we can say that the coefficients obey a power law decay if there exists constant C_1, $q > 0$, such that $|c_i| \le C_1 i^{-q}$. The larger q is, the faster the magnitudes decay, and the more compressible the signal [130].

2. Sensing matrix

A signal cannot be recovered from the compressed data if an incorrect sensing matrix is used. To guarantee the reconstruction of original signals, a sensing matrix must obey the Restricted Isometry Property (RIP). The definition of RIP is shown as below.

Matrix *A* satisfies the restricted isometry property of order *k* if there exists a $\delta_k \in (0,1)$ such that

$$(1-\delta_k)\|x\|_2^2 \le \|Ax\|_2^2 \le (1+\delta_k)\|x\|_2^2 \tag{2.17}$$

holds for all $x \in \sum_k$ [136]. RIP, as a norm, is widely used in constructing the sensing matrix.

Currently, it is proven that a random matrix satisfies the RIP with a high probability if the entries are chosen according to a Gaussian, Bernoulli, or more generally any sub-Gaussian distribution [130]. Besides the random matrix, a deterministic matrix can be used for the construction of the sensing matrix [141].

3. Signal recovery via ℓ_1 minimization

The best recovery method should be ℓ_0 minimization, because it can give the most sparse signal, but the algorithm is NP hard; therefore, ℓ_1 minimization is used as a replacement, which is shown in (2.18) [130].

$$x' = \underset{z}{\operatorname{argmin}}\|z\|_1 \qquad \text{subject to} \qquad z \in B(y) \tag{2.18}$$

where $B(y) = \{z: Az = y\}$.

For a noise-free signal recovery, we can use the following theorem to guarantee that the solution x' in (2.18) can very closely approximate the original vector *x*.

Suppose that *A* satisfies the RIP of order 2*k* with $\delta_{2k} < \sqrt{2} - 1$ and we obtain measurements of the form *y* = *Ax*. Then when $B(y) = \{z: Az = y\}$, the solution x' to (2.18) obeys (2.19) [130].

$$\|x' - x\|_2 \le C_0 \frac{\sigma_k(x)_1}{\sqrt{k}} \tag{2.19}$$

The specific recovery algorithms include the ℓ_1 minimization algorithm [142] and the Greedy algorithm [143] [144].

2.3.2.2 Neural Signal Processing Using Compressed Sensing Technique

In this section, we review some applications of neural signal processing based on the CS technique. Figure 2.19 illustrates the structure of analog and digital CS-based compression systems; currently, nearly all CS systems are designed as one of these two structures.

Currently, it is still in dispute whether neural signals are sparse or not. For some biomedical signals, some authors regard neural spikes as sparse in the wavelet domain [145] [146]; others

suggest that EEG signals can be compressed in the Gabor domain [9], and that ECG signals are sparse in the wavelet domain [147]. The author in [139] asserts that EEG signals are not sparse in the time or transformed domain. Therefore, it is appropriate to find a method to compress both sparse and non-sparse neural signals in the time domain.

As shown in Figure 2.19, the CS technique can be applied using analog and digital methods. Some applications are based on analog circuit implementation, such as the random demodulator [148] [149], random filtering [150], the modulated wideband converter [151], random convolution [152], and the compressive multiplexer [153]. Some of these analog applications are introduced below.

The block diagram of the random demodulator (RD) is shown in Figure 2.20 [149]. This structure includes a random number generator P_c, a mixer, an accumulator, and a sampler. Based on this block, the random demodulation pre-integrator (RMPI) (in some articles, it is called the modulated wideband converter) is proposed [148] [151]. This structure is composed of parallel channels of RD, as shown in Figure 2.21 [148]. RMPI can reduce the sampling rate of the system, but it needs more multiplexers or mixers.

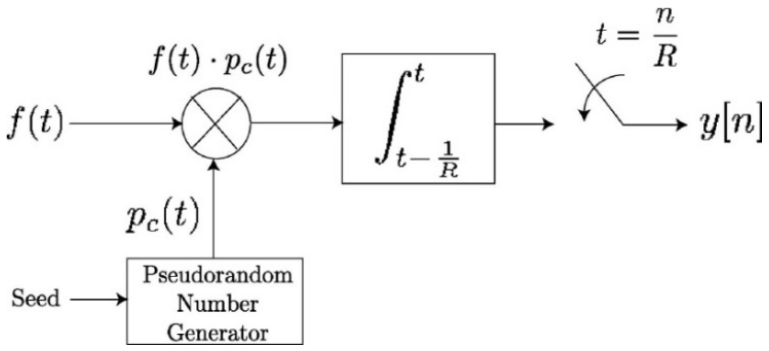


Figure 2.20 Block diagram of the random modulator [149]

To further reduce the power consumption of the RMPI, another structure, called spread spectrum random modulator pre-integrator (SRMPI), is proposed, shown in Figure 2.22 [148]. Compared with the RMPI, this structure uses another random block to randomly modulate the input signal. Compared using the traditional Nyquist sampling method, RMPI reaches 3% power reduction, and SRMPI reaches up to 43% energy saving [148].

The digital implementation of the RD is called the CS encoder, shown in Figure 2.23 [134]. This structure uses several multiplexers and adders to make linear transformations, and it also contains

a sensing matrix generator. The compression of the EEG signals reduces the power consumption of the whole system, reported as 1.9 μW at 0.6 V [134].

There are already some implementations of analog and digital circuits based on the CS theory, but which circuit implementation is superior – analog or digital – is still a matter of dispute [134] [148].

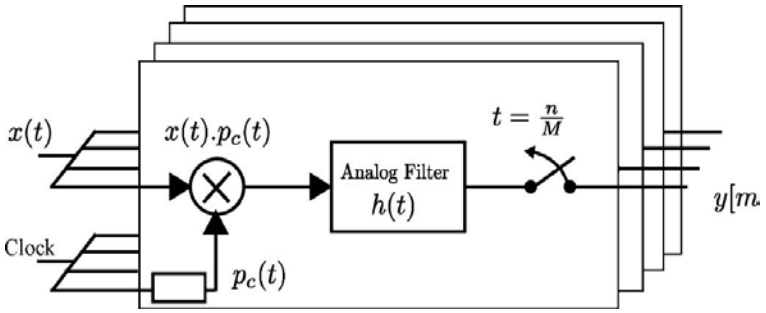


Figure 2.21 Block diagram of random demodulator pre-integrator (RMPI) [148]

An important consideration in all CS applications is how to generate the sensing matrix. As noted above, researchers agree that the sensing matrix can be constructed from a random matrix. In the following text, we review sensing matrix generation methods or circuits.

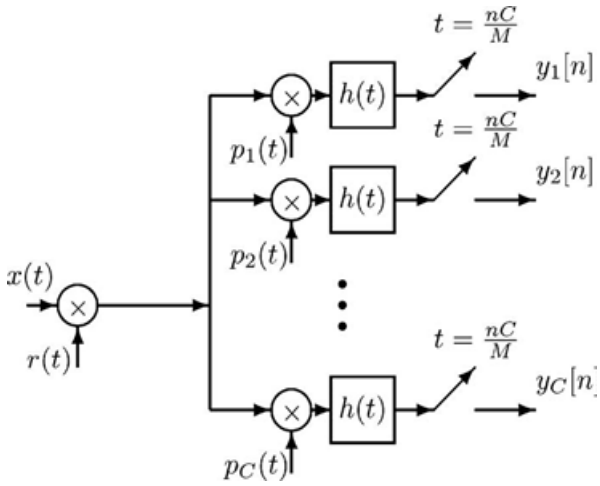


Figure 2.22 Block diagram of SRMPI [148]

The simplest method to generate the sensing matrix is using a look-up table or memory, but this method cannot be used for large measurements circuits [134].

In addition to this simple method, another method called pseudorandom number (PN, also called pseudo-random bit sequence (PRBS)) binary sequence can be used to generate the random matrix

[138]. This method is much more compact than the look-up table implementation, but the PN generator and associated clocks are the largest contribution to power consumption [154].

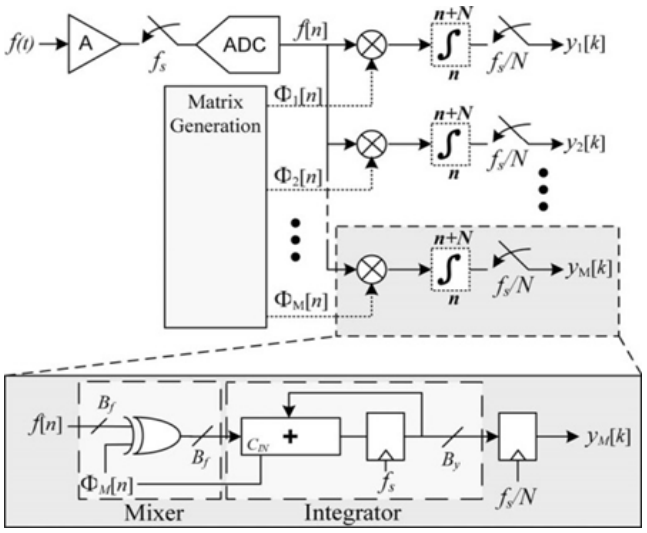


Figure 2.23 Block diagram of CS encoder [134]

To improve the PN generator, a double-PN generator is used. The structure is shown in Figure 2.24 [134]. This block uses two PRBSs to create the columns of the sensing matrix. This structure can reduce the whole power consumption by 10%. The entries of the PN generator obey the Bernoulli distribution, which are ±1 with the same probabilities.

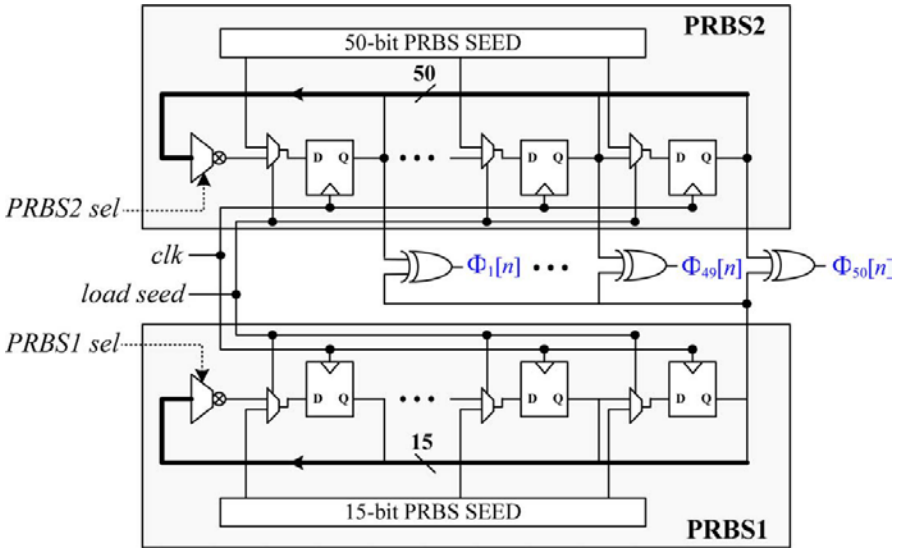


Figure 2.24 Block diagram of the measurement matrix generation block [134]

Moreover, there is a sensing matrix generating method called sparse binary matrices generator [8] [155]. This method is attractive; because it consists of 0/1 instead of ±1, when the number of the +1s in each column is a small fixed number, many of the calculations can be avoided.

Finally, several system designs using CS techniques are proposed for processing some biomedical signals. In [156], the authors proposed a data-dictionary-based signal processing system using CS, based on the similarity of shapes of the spikes. The proposed system is shown in Figure 2.25; in [11] and [64], the authors discussed the corresponding circuit design based on this system, which is shown in Figure 2.26. The area can be as little as 0.11 mm^2/channel and the power consumption 16 μW/channel (CMOS 0.18 μm, 20 kHz). The multichannel design and neural spike reconstruction are discussed in [157] and [158] respectively.

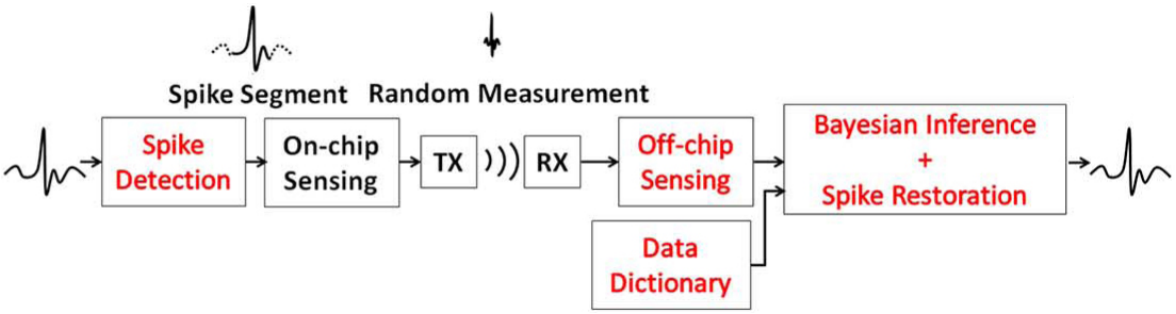


Figure 2.25 Proposed data dictionary based CS system [156]

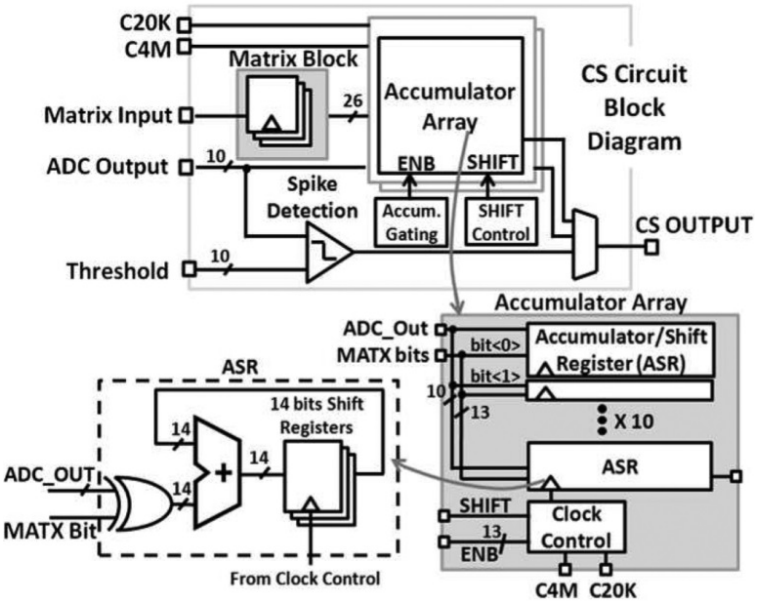


Figure 2.26 Proposed CS digital circuit [64]

In reference [159], the authors put forward a digital CS-based wireless sensor system. The authors in [138] proposed a 16-channel cortical recording system, with area and power consumption of 0.008 mm^2/channel and 0.95 μW/channel (CMOS 0.18 μm, 4 kS/s) respectively.

In summary, the CS technique has a simple structure which facilitates compression and implementation. Some designers already use this technique for the design of neural recording circuits; however, there are still some problems waiting to be solved. First, the sparsity of neural signals is not high in the time domain; therefore, a method to increase the sparsity of neural signals or to compress non-sparse or low-sparsity signals is required. Second, the design of the circuit based on the CS technique needs improvement, in areas including lowering the power consumption, shrinking the silicon area, and multichannel signal processing.

2.4 General Discussion of the Literature Review

In this section, we continue to discuss neural signal processing. Through reviewing the state-of-the-art compressed sensing techniques and automatic template matching-based spike detection and classification systems, we compare several similar works and explain the contribution of our research.

2.4.1 Neural Signal Processing Strategies

From the above literature review, it can be found that there are mainly two strategies for the neural signal processing inside a neural recording interface: signal reduction and compression. Firstly, signal reduction strategy involves spike detection [83] [106] and sorting [92] [123], and it is usually used for biomedical researches based on neural spikes. The advantage of this method is that it can largely remove the useless information of neural signals and retain their most important information. The disadvantage of this method is that this strategy causes distortion or loss of the data information. For example, for the neural spike detection, the data are obtained only as an impulse signal or in a time series which are no longer the signal itself [148]. Besides, if the thresholds of the detection are not properly set, then the spikes cannot be detected. The feature extraction usually requires a period of time to train, so the precision of this method usually cannot be guaranteed, and the hardware design of feature extractor is also complicated [160]. Second, for the signal compression strategy, the compressed sensing technique is a new signal compression technique. The advantage of the CS technique is that it can maximumly retain

the details of the recorded signals and has a simple structure, and disadvantage of this method is the limits of its usage, which is only used for sparse signals [130].

2.4.2 Discussion of Sensing Matrices

In 2.3.2, we briefly review the theory of the compressed sensing technique. From this section, it can be found that the traditional compressed sensing technique is mainly focused on the sparse signals, but in reality, not all of the signals are sparse, and using the approximation or changing the basis also cannot acquire sparse signals. Therefore, if signals are not sparse, the CS technique cannot be applied to compress them.

For the CS technique, the sensing matrix is an important research content, which has great influence on the signal compression and reconstruction [130]. The sensing matrix can be divided into random and deterministic matrices. In Table 2.3, we give a comparison between random and deterministic sensing matrices. The random matrices, such as the sub-Gaussian sensing matrix [134] [148] or the random discrete Fourier transmission matrix [161], are largely used by most of the designers. However, the random sensing matrix has disadvantages, for example, it usually needs a large amount of space to store the random matrices, and the superior randomness is usually needed for guaranteeing the compression performance. Besides, a random number generator usually has large power consumption and a large silicon area, which is not a good option for an implantable device [134].

Moreover, a deterministic sensing matrix is another option for the design of sensing matrix. There are several deterministic sensing matrices, such as the Discrete Chirp sensing matrix, the Reed Muller sensing matrix, the Bose-Chaudhuri-Hocquenghem sensing matrix, and low-density parity-check (LDPC) matrix. The advantage of the deterministic matrix is that it can generate the items of the sensing matrix on the fly without storing the data, and it is also easy to reconstruct original signals. However, current deterministic sensing matrices are very complicated in the hardware implementation, and they cannot be used for a non-sparse or low-sparse signal; although a low-density parity-check (LDPC) matrix contains only 0’s and 1’s, the compression of a non-sparse or low-sparse signal requires a very high-girth sensing matrix that is very difficult to generate [135] [162]. Therefore, a deterministic sensing matrix with simple structure and high compression performance needs to be researched.

Table 2.3 Comparison between random and deterministic sensing matrices

Type	Advantages	Disadvantages
Random sensing matrix	More mature in application	Needing a large space; needing a high-randomness random number generator
Deterministic sensing matrix	On the fly without storing data; reconstruction performance can be guaranteed	Complicated, not easy for implementation in hardware design

To explain the contribution and necessity of our research, In Table 2.4, we compared the performance of signal compression based on the CS technique using several state-of-the-art random or deterministic sensing matrices, such as the digital wavelet transform-based sensing matrix [8], Chirp sensing codes matrix [163], Bose-Chaudhuri-Hocquenghem sensing matrix [164], Ternary matrix [164], Elliptic curve matrix [141], Fourier-based transform sensing matrix [165]. From the comparison, several conclusions can be found. First, all the compared sensing matrices are used to compress sparse signals. In Table 2.4, it can found that the research objectives are all high-sparsity signals, and low-sparsity or non-sparse signals cannot be compressed through these sensing matrices. Second, the reconstruction error has a tight relationship with the degree of the sparsity of signals. Under a large compression rate, if the reconstruction error needs to be kept small, it requires the signals to have a large degree of sparsity. Besides, if the degree of sparsity is determined, using a small compression rate can reconstruct the original signal with a small reconstruction error. Third, the compared deterministic sensing matrices have a better compression performance than the random sensing matrices in Table 2.4. For signals with a similar degree of sparsity and having similar reconstruction error, using the deterministic sensing matrices can compress signals with a larger compression rate. Finally, from the comparison, two challenges for the construction of sensing matrices can be found:

1. Applying the CS to compress low-sparsity or non-sparse signals.

2. Compressing a compressible signal with a large compression rate while this signal can be reconstructed with high fidelity (very small reconstruction error).

Table 2.4 The signal compression performance of some compressed sensing matrices

Sensing matrix	**Types**	**Degree of Sparsity**		**Compression Rate (%)**		**Reconstruction Error**	
Chirp sensing codes matrix	Deterministic	0.95	0.90	98	98	≈ 0	> 1
Bose-Chaudhuri-Hocquenghem matrix	Deterministic	0.98	0.94	88	88	≈ 0	> 1
Ternary matrix	Deterministic	0.996	0.99	98	98	≈ 0	> 1
Elliptic curve matrix	Deterministic	0.99	0.97	93	93	≈ 0	> 1
Digital wavelet transform-based sensing matrix	Random	0.97	0.97	50	80	0.1	0.6
Fourier-based transform sensing matrix	Random	0.98	0.98	88	96	≈ 0	≈ 1
Our proposed sensing matrix	Deterministic	0	0.98	96	96	< 0.2	< 0.1

In chapter 3, we proposed a sensing matrix which tries to resolve these two issues. From the comparison in Table 2.4, our system can compress the sparse and non-sparse signals with a relatively large compression rate and a small reconstruction error. In this chapter, we proposed a sensing matrix which can compress the sparse and non-sparse signals with a large compression and a small reconstruction error. We use the similarity that is in a signal to make the compression, which can largely compress a specific neural signal may contain many identical (or similar) points, Additionally, we use the advantages of the deterministic sensing matrix to construct a sensing matrix that is based on the clustering of the neural signal itself, also the process of the construction is simple, which is appropriate for hardware implementation.

2.4.3 Discussion of Neural Signal Processing Systems

In section 2.3.2, we discussed several signal processing systems which are based on the CS techniques. In this section, we continue to discuss and compare these systems, and illustrate the challenges of signal processing for implantable neural recording devices.

From the comparison of different neural recording systems in Tables 2.1 and 2.2, it can be found that a neural signal processing system mainly has two main functions: detecting neural spikes and compressing neural spikes or original signals. In Table 2.5, we further compare several neural signal processing systems inside implantable neural recording devices. From the comparison, several conclusions can be acquired. First, current neural signal processing systems usually implement one signal processing strategy, which does not provide both functions of spike detection and signal compression. For some processing systems, they may include spike detection and compression, but the compression block is only to effectively transmit the detected neural spikes, and it does not include any processing for lossless compression of original neural signals. Second, for the CS-based neural signal processing systems in Table 2.5, it is mainly used to compress sparse signals, and none of signal processing systems are designed to compress non-sparse signals, but efficiently compressing the non-sparse signals can enlarge the scope of application for neural signal processing systems; therefore, it is necessary to design a CS-based signal processing system for low-sparsity or non-sparse signals. Third, some signal processing systems only include single channel processing, and with the increasing of recorded channels and high requirement of the users, efficient multichannel neural signals processing is important and significant, so multichannel signal processing still needs to be researched. Finally, power consumption and area are two very important parameters for an implantable neural signal processing system, and it is necessary to research some methods to reduce both parameters while maintain the best processing performance of the processor.

In chapter 5, we implement a neural signal processing system, which includes spike detection and CS-based neural signal compression. In this system, the signal compression block is based on our proposed MDC matrix, so the sparse and non-sparse signals can both be compressed through the implemented processing system. Moreover, our proposed system can effectively deal with the signals recorded by single-channel and multichannel recording devices, and also, from the

comparison in Table 2.5 our proposed processing system has relatively small area and low power consumption while maintains good processing performance.

Table 2.5 Comparison of several signal processing systems for neural recording devices

Reference	**[166]**	**[167]**	**[168]**	**[134]**	**[64]**	**[138]**	**Our system**
Technology (μm CMOS)	0.5	0.5	0.065		0.18	0.18	0.13
Signal reduction	Spike detection	Spike detection	Spike detection feature extraction	-	-	-	Spike detection
Signal compression	Discrete wavelet transform-based	-	-	CS-based	CS-based	CS-based	CS-based
Signal for compression	-	-	-	Sparse signals	Sparse signals	Sparse signals	Sparse and non-sparse signals
Number of channels (electrode)	32	32	16	1	32	16	256
Area per channel (electrode) (mm^2)	0.18	0.12	0.07	0.103	0.11	0.008	0.03
Power consumption per channel (electrode) (μW)	95	75	4.68	1.9	0.83	0.95	12.5

2.4.4 Discussion of Spike Detection Methods

In section 2.3.1, we review the categories of the spike detection and sorting methods. In this section, we continue to discuss the methods of spike detection and classification, and the research issues of automatic template matching-based spike detection system.

The main purpose of the spike detection and classification is correctly detecting neural spikes from the recorded neural signals and separating the spike series from the composite spikes. From

the above literature review, it can be found that spike detection can be divided into amplitude-based, energy-based and template matching methods. In Table 2.6, we compare the performance of these three methods. The idea of the first two methods is that when a spike occurs, the signal usually has a sudden change in amplitude or energy [97] [103] . Comparing with the template matching method, both methods cannot make the spike classification and they have poor detection accuracy for signals with a low signal-to-noise ratio. The template matching method applies another idea for spike detection, that is, the designer can use a detected spike (spike template) to compare with recorded neural signals, and then locate neural spikes [112]. This method has a high accuracy, but this method usually has high complexity, which is not easy for implementation. Comparing all of three methods, using template matching method is better in the detection accuracy, but the disadvantage of this method is its complexity and the requirement of the spike templates, therefore, it is necessary to research a low-complexity template matching-based spike detection method without foreknowing templates.

Table 2.6 Comparison among amplitude-based, energy-based and template matching-based spike detection

Type	**Advantages**	**Disadvantages**
Amplitude-based detection method	More mature in application; easily implemented	Cannot make the spike classification; poor accuracy for low-SNR signals
Energy-based detection method		
Template matching	High detection accuracy for low SNR signals; can make the spike classification	Complicated, not easy for hardware implementation

In section 2.3.1.3, we review several template matching-based spike detection systems. Based on the review, it can be found that for the template matching-based spike detection methods, the complexity and detection accuracy are two important factors. First, designing a system with low complexity can reduce the difficulties of hardware design, especially for the implantable neural recording devices. Second, the designed system should have high detection accuracy which is the

basis of many biomedical researches. In Table 2.7, we compare the complexity and detection accuracy of several automatic template matching-based spike detection systems. From the comparison, it can be found that the complexity of system can be further reduced, and detection accuracy also can be improved.

Table 2.7 Comparison of several template matching-based spike detection systems

Reference	**Detection method**	**Types**	**Complexity**	**Accuracy**
[112]	Fast normalized correlator	Automatic	(N+1)*multiplication + (N+1)*addition + 1*division + 1* squared root[a]	TPR : 0.84(SNR =3) FPR : 0.01(SNR =3)
[115]	M-sorter	Non-automatic	> (N*multiplication + (N+1)*addition)	TPR : ≈ 0.85(SNR > 6) FPR : ≈ 0.2 (SNR > 6)
[169]	Wavelet-based template matching	Non-automatic	> (N*multiplication + (N+1)*addition)	TPR : 0.6(SNR = 3) FPR : 0.04(SNR = 3) TPR : 0.9(SNR = 5) FPR : 0.04(SNR = 5)
[170]	Deconfusion method	Automatic	P* (Number of neuron)^2 * length of the filter[a]	TPR : 0.86(SNR =3) FPR : 0.04(SNR =3)
Our system	Bayesian inference-based	Automatic	N*multiplication + (N+1)*addition	TPR : 0.90(SNR=3) FPR : 0.04(SNR=3) TPR : 0.90(SNR=6) FPR : 0.03(SNR=6)

Moreover, for the template matching-based spike detection method, it can be divided into non-automatic and automatic template matching. Non-automatic template matching means that the templates need to be given in advance and template matching means that the templates can be generated by the device itself. Currently, designing an automatic template matching system is necessary [112]. For an automatic template matching system, when the templates are not known, it needs to generate the templates first and the spike sorting needs to be involved. To generate the spike templates, three are three main steps: spike alignment, feature extraction and spike classification. For the spike alignment, there are two mean methods: aligning each spike to the point of its maximum amplitude or the point of maximum slope [88]. Besides, there are several feature extraction techniques, such as principal components analysis [171], discrete wavelet transform [172], matched subspace detector [173], etc. In [174] , a method, called discrete derivatives (DD) method, is described as less complicated in terms of calculation while

maintaining fairly high accuracy, and it is suitable for use in the general circuit design. Spike clustering is the final step to sort out detected spikes from different neurons. The K-means method is a sophisticated method for the spike clustering, but it needs to manually set *k* in order to determine the number of required clusters [128] [175]. The Osort algorithm, introduced in section 2.3.2.1, can automatically determine *k*, which can be used for the automatic template matching system [116].

From above literature review, it can be found that to design template matching-based spike detection system, it has two main challenges. First, it needs to design a low-complexity and high-detection accuracy system, and second, the templates need to be automatically generated. In chapter 4, we propose an automatic template matching-based spike detection and classification system. From the comparison in Table 2.7, the system has a simple structure and fast calculation, which is appropriate for the hardware design. Besides, our proposed system can automatically generate templates and perform spikes detection and classification.

CHAPTER 3 ARTICLE 1 : NEURAL SIGNAL COMPRESSION USING A MINIMUM EUCLIDEAN OR MANHATTAN DISTANCE CLUSTER-BASED DETERMINISTIC COMPRESSED SENSING MATRIX

According to the discussion in chapter 2, signal compression is an important signal processing method for implantable neural recording interfaces. Among different signal compression methods, compressed sensing technique is a new technique for signal compression, which can be employed to compress neural signals. For the traditional compressed sensing theory, it is mainly focus on the sparse signals. However, neural signals are usually not sparse in the time domain but contain lots of similar non-zero points; therefore, it is necessary to research a method to compress low-sparsity and non-sparsity signals. Besides, it needs to research a method to compress signals with a large compression rate and a mall reconstruction error.

In this chapter, we put forward a new method to compress not only a sparse signal but also a non-sparse signal that has identical points. Firstly, several concepts about the identical items of the signal are introduced; then, the method to construct the Minimum Euclidean or Manhattan Distance Cluster-based (MDC) deterministic compressed sensing matrix is given. Moreover, the Restricted Isometry Property of the MDC matrix is proved. Thirdly, three groups of real neural signals are used for the validation. Six different random or deterministic sensing matrices under diverse reconstruction algorithms are used for the simulation. From the simulation results, it can be proved that the MDC matrix can largely compress neural signals and also have a small reconstruction error. For a six-thousand-point signal, the compression rate can be up to 98%, whereas the reconstruction error is less than 0.1. In addition, from the simulation results, the MDC matrix is optimal for a signal with a long length. Finally, the MDC matrix can be constructed by zero and one; also, it has a simple construction structure, which is very practicable for the design of an implantable neural recording device.

(Biomedical Signal Processing and Control, publication date: May 2015)

Neural Signal Compression Using a Minimum Euclidean or Manhattan Distance Cluster-Based Deterministic Compressed Sensing Matrix

Nan Li [a*], Mohamad Sawan[a]

[a] Polystim Neurotechnologies Lab.

Electrical Engineering Dept., Polytechnique Montreal

2900 Edouard-Monpetit, H3T 1J4, Montréal (QC), CANADA

ABSTRACT — Multichannel wireless neural signal recording systems are a prominent topic in biomedical research, but because of several limitations, such as power consumption, the device size, and enormous quantities of data, it is necessary to compress the recorded data. Compressed sensing theory can be employed to compress neural signals. However, a neural signal is usually not sparse in the time domain and contains a large number of similar non-zero points. In this article, we propose a new method for compressing not only a sparse signal but also a non-sparse signal that has identical points. First, several concepts about the identical items of the signal are introduced; thus, a method for constructing the Minimum Euclidean or Manhattan Distance Cluster-based (MDC) deterministic compressed sensing matrix is given. Moreover, the Restricted Isometry Property of the MDC matrix is supported. Third, three groups of real neural signals are used for validation. Six different random or deterministic sensing matrices under diverse reconstruction algorithms are used for the simulation. From the simulation results, it can be demonstrated that the MDC matrix can largely compress neural signals and also have a small reconstruction error. For a six-thousand-point signal, the compression rate can be up to 98%, whereas the reconstruction error is less than 0.1. In addition, from the simulation results, the MDC matrix is optimal for a signal that has an extended length. Finally, the MDC matrix can be constructed by zeros and ones; additionally, it has a simple construction structure that is highly practicable for the design of an implantable neural recording device.

Keywords — multichannel neural recording device, low power design, neural signal processing and compression, deterministic compressed sensing matrix, restricted isometry property.

3.1 Introduction

Over the past several years, neural recording and stimulation systems have contributed substantial benefit to patients who suffer from Parkinson's disease, major depressive disorder, and epilepsy [176], [177]. However, research and applications demand an increasing number of requirements, which implies more requirements for the neural recording system. These requirements include having high-density integration of the recording electrodes [58] [59] (now, to our knowledge, a neural recording system can integrate more than a thousand electrodes [61]), low temperature (an increase in the temperature of the cortex must be smaller than one centigrade, which means that the maximum power density should be 0.8 mW/mm^2 for the exposed tissue area [178]), long device lifetime, and small device size. Among all of these requirements, the power consumption is one of the most challenging issues. In a patient who requires an implantable medical device, there must be limit to the frequency of replacing the batteries to both reduce the cost of the surgeries and improve the quality of life. For example, if there is a portable battery that has an energy density in the range of 1 W-hr/cc, a battery volume on the order of 10 μW average power per cubic centimeter is required for a 10-year device life span [134]. Moreover, many of the implantable devices integrate a wireless transmission part, which aggravates the situation of having stringent energy constraints, because large amounts of recorded data required a very high carrier frequency, which substantially increases the power consumption of the device [57] [72] [179]. A common ultra-wideband (UWB) radio exhibits energy-efficiencies in the nJ/bit range, whereas the power consumption of the other components is 10^3 times less than that of the UWB radio [134]. Therefore, a signal reduction strategy for an implantable device should be employed to minimize the power consumption of the system.

Most of existing methods for implementing integrated data reduction under these constraints involves detecting neural spikes [83] [106] or extracting the data features of the signal [92] [123]. However, both of these methods cause distortion or loss of the data information. For example, in a neural spike-detection recorder, the data are obtained only in a time series or as an impulse signal but not as the signal itself [148]. If the thresholds of the detection are not properly set, then the spikes cannot be detected. At the same time, the feature extraction requires a period of time to train. Based on this method, the precision usually cannot be guaranteed, and the hardware design is also complicated [160]. Therefore, we must find a new method that does not lose the details of the signal to accomplish the goal of recording the signal.

Compressed sensing (CS) technology gives us a new choice for signal compression. In recent years, this approach has attracted considerable attention in the areas of computer science, applied mathematics and electrical engineering [131] [140]. CS technology can be divided into three main parts: sparse signal, signal reconstruction and sensing matrix.

3.1.1 Sparse Signal

CS theory is based on the sparsity of the signal. If a signal Y, which can be found in a basis such as $V = [v_1 , v_2 , v_3, \cdots, v_n]$ has a sparse representation, then the signal is called a sparse signal. Specifically, suppose Y can be described as in (3.1).

$$Y = VX \quad \text{or} \quad Y = \sum_{i=1}^{n} x_i v_i \tag{3.1}$$

where x_i is the coefficient vector for Y under the basis V. If Y is sparse, then the coefficient x_i must be almost zero or negligible, and as a result, they can be omitted without any loss.

If a signal is sparse under some basis, then it can be regarded as a compressible signal. Usually, a signal is not sparse, but if the basis can be changed, then the sparse representation under the new basis can be obtained. For example, a sine wave is not sparse in the time domain, but it is sparse in the Fourier domain.

3.1.2 Signal Reconstruction

There are many reconstruction methods; an example is the ℓ_1 (or ℓ_2) norm-based reconstruction method, which searches for the minimum ℓ_1 (or ℓ_2) value to construct the signal [136] [180]. This type of algorithm includes the basis pursuit algorithm (BP), matching pursuit algorithm (MP), orthogonal matching pursuit algorithm (OMP) [181] [182], and threshold-based method (such as the iterative hard or soft thresholding algorithm [182] [183]). Probability-based reconstruction methods constitute another type; for example, the sparse Bayesian method uses the maximum likelihood to reconstruct the signal [184] [185]. As of now, it has been proven that for a k-sparse signal, if the order of the measurement is $2k$, the original signal can be recovered exactly [186].

3.1.3 Sensing Matrix

Not all of the signals are sparse, and the “sparse” basis is usually difficult to find. Although the “sparse” basis of a signal can be found, how to implement it into a device is still difficult [9]. To

compress the non-sparse signal, we introduce a new concept for the compressed sensing, which is that not only the zero points in a signal can be compressed but also the identical non-zero points in the signal can be compressed. Therefore, in this article, we construct a deterministic sensing matrix that is based on this idea to compress the neural signals.

The sensing matrix can be divided into two types: random and deterministic matrices. Currently, most of the designers use a type of random matrix as a sensing matrix in the system, such as the sub-Gaussian sensing matrix [134] [148] or the random discrete Fourier transmission matrix [161]. However, the random matrix has disadvantages. First, storing the random matrix requires a large amount of space, and the effectively proven random sensing matrices require items with superior randomness, which causes there to be stringent requirements for the design of a random number generator. Moreover, a random number generator aggravates the complexity of the hardware design, especially for an implantable device, because the generator usually has large power consumption and a large silicon area. Therefore, the current random sensing matrices are not the best choice for an implantable hardware design.

In addition, a deterministic sensing matrix is discussed as an optional type of sensing matrix. The advantage of the deterministic matrix is that it can generate the items of the sensing matrix on the fly without storing the data, and it is also easy to reconstruct the original signal. However, current deterministic sensing matrices, such as the Discrete Chirp sensing matrix [163], the Reed Muller sensing matrix [187], and the BCH sensing matrix [164], are also complicated with respect to the hardware implementation, and they cannot be used for a non-sparse or low-sparse signal; although a low-density parity-check (LDPC) matrix contains only 0’s and 1’s, the compression of a non-sparse or low-sparse signal requires a very high-girth sensing matrix that is very difficult to generate [135] [162]. Therefore, a novel deterministic sensing matrix must be constructed.

Moreover, there are two important contributions in this article. First, we use the similarity that is in a signal to construct the compression. In fact, a specific neural signal may contain many identical (or similar) points, and traditional compressed sensing concerns only the zero items in a signal; it does not concern two identical (or similar) non-zero points in the signal. Therefore, we research these identical or highly similar non-zero points, i.e., the similarity of the points in a signal, from the perspective of compressed sensing theory. Additionally, we use the advantages of the deterministic sensing matrix to construct a sensing matrix that is based on the clustering of

the neural signal itself. In brief, the primary contribution of this article is that we design a deterministic compressed sensing matrix to compress non-sparse or low-sparse signals that have identical non-zero points, and the compressed signals can be largely recovered.

To illustrate our work, we give definitions and proof for the MDC sensing matrix in section 3.2. We introduce the dataset of the simulation in section 3.3. The simulation results and a discussion based on the MDC sensing matrix are given in section 3.4. Finally, in section 3.5, we provide a conclusion.

3.2 Minimum Euclidean or Manhattan Distance Cluster-Based Deterministic Sensing Matrix

First, we provide the definitions of several basic concepts and the method of MDC matrix construction. (Some important variables or symbols are illustrated in Table 3.1).

The most important concept in compressed sensing theory is the Restricted Isometry Property (RIP), which is shown as follows.

Restricted Isometry Property An $M \times N$ sensing matrix Φ is said to satisfy the Restricted Isometry Property of order k if it satisfies (3.2):

$$(1 - \varepsilon_k)\| X \|_2^2 \leq \| \Phi \mathrm{X} \|_2^2 \leq (1 + \varepsilon_k)\| X \|_2^2 \tag{3.2}$$

for all of the *k-sparse* vectors X. The restricted isometry constant ε_k of matrix Φ lies between 0 and 1. The restricted isometry constant ε_k, $k \in (1, n)$ of sensing matrix Φ is defined as (3.3):

$$\varepsilon_k(\Phi) = \max_{|T| \leq k} \left\| \Phi_T^* \Phi_{\mathrm{T}} - I_{\mathbb{R}^T} \right\| = \max_{|T| = \lfloor k \rfloor} \left\| \Phi_T^* \Phi_{\mathrm{T}} - I_{\mathbb{R}^T} \right\| \tag{3.3}$$

where the maximum is over all subsets $T \subseteq [n]$ with $|T| \leq k$ or $|T| = \lfloor k \rfloor$, and Φ_T means all $M \times k$ sub-matrices of Φ.

After the presentation of the RIP, we give some basic concepts to construct the MDC matrix.

Definition 1: (Equal Index Permutation) Given a vector $X(x_1, x_2, \cdots, x_n)$, there exists a permutation $A_1(a_1, a_2, \cdots, a_t)$ of the index vector $(1, 2, \cdots, n)$, and there is a vector that is based on this index permutation $X_{A_1}(x_{a_1}, x_{a_2}, \cdots, x_{a_i}, \cdots, x_{a_t})$, $x_{a_i} \in X$. If every two items from X_{A_1} are identical under some measures, in other words, $x_{a_i} = x_{a_j}$, $x_{a_i}, x_{a_j} \in X_{A_1}$, A_1 is called an equal index

permutation. If this measure is based on Euclidean (or Manhattan) distance, then A_1 is called an equal index permutation under the Euclidean (or Manhattan) distance.

Table 3.1 Symbols and variables

Variable or notation	**Meaning**	**Variable or notation**	**Meaning**
C	One cluster C	D(K)	Degree of the sparsity; $D(K) = 1 - (k/N)$
→	Approximate to	K *or* k	Sparsity of the signal
Y	Measurement; $Y = \Phi X$	$\Phi_{M \times N}$	M rows N columns sensing matrix $\Phi = [\phi_1 ; \phi_2 ; \cdots ; \phi_m]$
L(X)	Length of a vector X	I(C)	Size of a cluster C
CR	Compression rate; $CR = 1 - (N/M)$	Set(C) or S	Cluster set; Set(C) contains n clusters
MD	Maximum distance; The maximum value between two points. There are two different maximum distances in this article: inner MD and 0-MD. The inner MD is mainly the distance between two points of a vector. The 0-MD indicates the distance between one point with the zero-value point in a vector.	RER	Reconstruction error; If $\nabla(\Phi x)$ is the reconstruction of a measurement, so the reconstruction error is the Euclidean distance between the reconstructed signal and the original signal $RER = \|\nabla(\Phi x) - x\|_2 / \|x\|_2$

Table 3.1 Symbols and variables (cont'd)

Variable or notation	**Meaning**	**Variable or notation**	**Meaning**
CEER	Compression error of the expected measurement; CEER $= \|\|\mathrm{E}(\|\|\Phi x\|\|_2^2) - \|\|x\|\|_2^2\|\|_2 / \|\|x\|\|$	$\delta(S)$	Standard deviation of the size of all of the clusters in a cluster set, in other words, $\delta(\mathrm{I}(C(x_{A_1})), \mathrm{I}(C(x_{A_2})), \cdots, \mathrm{I}(C(x_{A_n})))$
$\mathrm{I}_{\max}(\mathrm{Set}(\mathrm{C}))$	Maximum size of a cluster in a cluster set	CER	Compression error; $\mathrm{CER} = \|\|\|\|\Phi x\|\|_2^2 - \|\|x\|\|_2^2\|\|_2 / \|\|x\|\|_2^2$
$\mathrm{R}(S)$	$\mathrm{R}(S) = \mathrm{I}_{\max}(S) \,/\, (N/M)$	$\mathrm{R}(K,M,N)$	$\mathrm{R}(K,\ M,\ N) = (k - M)/N$

A vector that has identical items can be clustered into several clusters according to the minimum Euclidean or Manhattan distance; thus, some other concepts are given.

Definition 2: Given a vector $X\,(x_1, x_2, \cdots, x_n)$, there exists an index set that contains M equal index permutations under the Euclidean (or Manhattan) distance, i.e., $A_M\,(A_1, A_2, \cdots, A_m)$, where $A_i = (a_{i_1}, a_{i_2}, \cdots, a_{i_j}, \cdots, a_{i_t})$, $a_{i_j} \in (1, 2, \cdots, n)$. X can be clustered into M clusters according to the index A_M, in other words, $X_{A_M}(x_{A_1}, x_{A_2}, \cdots, x_{A_m})$. If $a_{1_i} \in (1, 2, \cdots, n)$

1. $\forall x_i \in X$, $x_i \in x_{A_i}$ and $x_i \notin x_{A_j}$, A_i, $A_j \in A_M$; $A_i \neq A_j$

2. $\forall x_i$, $x_j \in X$, $x_i \in x_{A_i}$, $x_j \in x_{A_j}$ and $x_i \neq x_j$, A_i, $A_j \in A_M$; $A_i \neq A_j$

Thus, A_M is called an exclusive equal index permutation set, and vector X is called an M-cluster exclusive vector under permutation set A_M.

Definition 3: (p-dissimilar vector) Assume that a vector $X\,(x_1, x_2, \cdots, x_n)$ is an M-cluster exclusive vector under the permutation set $A_M\,(A_1, A_2, \cdots, A_m)$, where $A_i = (a_{i_1}, a_{i_2}, \cdots, a_{i_j}, \cdots, a_{i_t})$, $a_{i_j} \in (1, 2, \cdots, n)$. According to definition 2, X can be clustered into M clusters based on the index A_M, in other word, $X_{A_M}(x_{A_1}, x_{A_2}, \cdots, x_{A_m})$. Let $p = M$; then vector X is called a p-dissimilar vector. The size of each cluster $C_{x_{A_i}}$ is $\mathrm{I}(C_{x_{A_i}}) = t$, $X_{A_i} \in X_{A_M}$. So $C_{x_{A_i}}$ is called a t-large cluster. If $t = 1$, then $C_{x_{A_i}}$

is called a unit-large cluster. If A_i, $\forall A_i \in A_M$, is an equal index permutation under the Euclidean (or Manhattan) distance, then X is called a minimum Euclidean (or Manhattan) distance p-dissimilar vector.

From the definition above, $t \neq 0$, because a cluster contains at least one point.

Lemma 1: Assume that there is a p-dissimilar vector X under permutation set $X_{A_M}(x_{A_1}, x_{A_2}, \cdots, x_{A_m})$ and that its length is L(X) = n; thus it can be clustered into M clusters, i.e., Set(C) = $\{C(x_{A_1}), C(x_{A_2}), \cdots, C(x_{A_m})\}$. Thus, Set($C$) satisfies (3.4) and (3.5).

$$\mathrm{I}(C(x_{A_1})) + \mathrm{I}(C(x_{A_2})) + \mathrm{I}(C(x_{A_3})) + \cdots + \mathrm{I}(C(x_{A_m})) = n \tag{3.4}$$

And

$$C(x_{A_i}) \cap C(x_{A_j}) = 0 \tag{3.5}$$

Definition 4: (*Equivalent Index Subset Vector*) Assume that there are two vectors, $X\ (x_1, x_2, \cdots, x_n)$ and $Y\ (y_1, y_2, \cdots, y_n)$, that have the same length L(X) = L(Y) = n. X is an M-cluster exclusive vector under permutation set $A_M\ (A_1, A_2, \cdots, A_m)$, where $A_i = (a_{i_1}, a_{i_2}, \cdots, a_{i_j}, \cdots, a_{i_t})$, $a_{i_j} \in (1, 2, \cdots, n)$. For a determined subset A_i, $A_i \in A_M$, if $\{y_{a_i} = r \mid a_i \in A_i\}$ and $\{y_{a_i} = 0 \mid a_i \notin A_i\}$, then Y is called an equivalent index subset vector of the vector X.

When $r = 1$, Y is called the unit equivalent index subset vector. When $r = 1/\sqrt{\| Y \|_0}$, it is called the normalized equivalent index subset vector. $\| Y \|_0$ is the total number of non-zero elements in vector Y. For a p-dissimilar M-cluster exclusive vector X, there are M equivalent index subset vectors.

Lemma 2: Assume that X is an p-dissimilar vector under permutation set $A_M\ (A_1, A_2, \cdots, A_m)$ and that its length is L(X) = n; then, it can be clustered into M clusters under an exclusive equal index permutation set, and the equivalent index subset vector of every cluster is $Y_M\{Y_1, Y_2, \cdots, Y_m\}$, which implies that it satisfies (3.6).

$$Y_i * Y_j^{'} = 0\ ,\ Y_i, Y_j \in Y_M \tag{3.6}$$

Proof: If $Y_i * Y_j^{'} \neq 0$, then $C(x_{A_i}) \cap C(x_{A_j}) \neq 0$, but the opposite holds under the assumption in lemma 1, so $Y_i * Y_j^{'} = 0$.

With the definitions above, we can construct the sensing matrix for a minimum Euclidean (or Manhattan) distance *p*-dissimilar vector.

Definition 5: (Minimum Euclidean or Manhattan distance cluster-based deterministic sensing matrix (MDC matrix))

If a vector X is a minimum Euclidean (or Manhattan) distance *p*-dissimilar vector, then we can construct a deterministic sensing matrix through the following three steps.

(St1) Divide X into M dissimilar clusters $\{C(x_{A_1}), C(x_{A_2}), \cdots, C(x_{A_m})\}$ based on the exclusive equal index permutation set $X_{A_M}(x_{A_1}, x_{A_2}, \cdots, x_{A_m})$.

(St2) The equivalent subset index vector of these clusters $\{C(x_{A_1}), C(x_{A_2}), \cdots, C(x_{A_m})\}$ is $\{\phi_1, \phi_2, \cdots, \phi_m\}$, $m \in \mathbb{N}$.

(St3) Composing the matrix with $\{\phi_1, \phi_2, \cdots, \phi_m\}$, $m \in \mathbb{N}$, which is $\Phi = [\phi_1 ; \phi_2 ; \cdots ; \phi_m]$.

Thus, Φ is called a minimum Euclidean or Manhattan distance cluster-based deterministic sensing matrix (MDC) matrix. If all of the ϕ_i , $i \in [1, m]$ in the Φ are the normalized equivalent index subset vectors, so Φ is called a normalized MDC (NMDC) matrix . If all the ϕ_i , $i \in [1, m]$ in the Φ are the unit equivalent index subset vectors, so Φ is called a unit MDC (UMDC) matrix. An example of the construction is shown as follows.

Given a vector $X(x_1, x_2, \cdots, x_6)$ and that X can be clustered into three clusters based on the minimum Euclidean (or Manhattan) distance (these clusters are $\{\{x_1, x_2, x_5\},\{x_3, x_6\},\{x_4\}\})$; therefore, the NMDC matrix for the vector X is Φ , which is shown in (3.7).

$$\Phi = \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 0 & 0 & 1/\sqrt{3} & 0 \\ 0 & 0 & 1/\sqrt{2} & 0 & 0 & 1/\sqrt{2} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.7}$$

To research the property of the MDC matrix, we give a similar definition based on the Restricted Isometry Property.

Definition 6: (*Cluster Restricted Isometry Property (CRIP))* An $M \times N$ sensing matrix Φ is said to satisfy the Cluster Restricted Isometry Property of order k if it satisfies (3.8):

$$(1 - \varepsilon_k)\| X \|_2^2 \leq \| \Phi X \|_2^2 \leq (1 + \varepsilon_k)\| X \|_2^2 \tag{3.8}$$

for all of the *k*-sparse *p*-dissimilar vectors X that construct Φ through definition 5.

There are two important points that must be explained. First, every vector X has its own MDC matrix, and the MDC matrix is a collection of all of the matrices that are built through the algorithm in definition 5. In the following sections, we show that the MDC matrix obeys the RIP for its related vector under some prerequisites. Moreover, the vector X is random, but if the clustering method and the measure are determined, then the MDC matrix of a certain vector X is determined; thus, the MDC can be regarded as a deterministic matrix.

Lemma 3: The NMDC matrix is a unit tight (or Parseval) frame.

Proof: Given a random vector $X \in \mathbb{C}^N$, and that the NMDC matrix Φ satisfies (3.9).

$$(\Phi * \Phi^{\mathrm{T}}) = \sum_{i,j=1}^{M} \phi_i \phi_j = I_M \tag{3.9}$$

Vector X satisfies (3.10).

$$\sum_{i=1}^{N} |\langle X, \phi_i \rangle|^2 = X * \Phi * \Phi^{\mathrm{T}} * X^{\mathrm{T}} = \|X\|_2^2 \tag{3.10}$$

Therefore, the NMDC matrix is a unit tight frame.

Lemma 4: The NMDC matrix satisfies the Cluster Restricted Isometry Property definitely.

Proof: From reference [130], for signal X, if the restricted isometry constant of Φ is ε_{2k} and $\varepsilon_{2k} < \sqrt{2}$ - 1, the solution to the ℓ_1 problem is a unique k-sparse solution.

Given a k-sparse p-dissimilar signal, the index set of the non-zero items is T. A new signal $X'$is constructed by the non-zero items $X'_{2T} \sim \{X_T, X_T\}$. The NMDC matrix of X' is Φ'. From Lemma 3, the NMDC matrix is an unit tight frame, in other words, $\|\Phi' X'\|_2 = \|X'\|_2$, which means that $\exists\, \lambda > 0$, $|\varepsilon_{2k}| < \lambda$; therefore, $X'_{2\mathrm{T}}$ can be recovered through the compression. Moreover, a subset T of the set $2T$ can be found to construct Φ, in other words, $\|\Phi_T X_T\|_2 = \|X_T\|_2$, which means that $\|\Phi x\|_2^2 = \|x\|_2^2$; thus, from measurement Y, it can exactly reconstruct X.

The NMDC matrix can compress any signal that contains identical items without considering the signal to be a sparse signal or not, because the NMDC matrix is a unit tight frame for its corresponding compressed signal; in other words, the restricted isometry constant ε_k of the NMDC matrix is 0. In the simulation, we can find that the UMDC (the items of the UMDC matrix are 0's and 1's) can reconstruct the original signal correctly, which indicates that UMDC also satisfies the Cluster Restricted Isometry Property.

Although the MDC matrix obeys CRIP, we still want to know whether the MDC matrix satisfies RIP, and we also hope to use the UMDC matrix that contains only zeros and ones. As a result, we prove that the UMDC matrix obeys RIP when $(k-M)/N \to 0$ and $\mathrm{I}_{\max}(\mathrm{Set}(C)) \le N/M$.

Theorem 1 [188]: Let $\Phi_{M\times N}$ be a sensing matrix, and let a vector X $(x_1, x_2, \cdots, x_n)$ be a random vector. Given the following inequality (3.11):

$$\Pr(|\|\Phi x\|_2^2 - \|x\|_2^2| \ge \varepsilon\|x\|_2^2) \le 2\mathrm{e}^{-Mc_0(\varepsilon)} \tag{3.11}$$

where $\varepsilon \in (0,1)$, and $c_0(\varepsilon) > 0$ is a constant that depends only on ε. If (3.11) is satisfied, then $\Phi_{M\times N}$ satisfies the concentration inequality.

From reference [188], it can be learned that if we assume a sensing matrix $\Phi_{M\times N}$ and a k-sparse signal and if Φ satisfies two conditions: $\mathrm{E}(\|\Phi x\|_2^2) = \|x\|_2^2$ and $\|\Phi x\|_2^2$ converges to $\|x\|_2^2$, then Φ obeys the RIP with a probability of at least $1 - 2\exp(-c_0(\varepsilon/2)M)(12/\varepsilon)^k, \varepsilon \in (0,1)$. In Lemmas 5 and 6, we prove both conditions.

Lemma 5: We are given a k-sparse n-length p-dissimilar random vector $X(x_1, x_2, \cdots, x_n)$ and that every item of X is uniformly distributed in the vector X, and it can be clustered into M t-clusters. Every two items in the vector have the same probability of being identical. The number of non-zero items in all of the clusters are $\{l_1, l_2, \cdots, l_m\}$, and their sum is $\sum_{i=1}^{M} l_i = k$. $\Phi\{\phi_1, \phi_2, \cdots, \phi_m\}$ is an $M \times N$ UMDC matrix, and $Y = \Phi X$. The ℓ_1 norm of each column $|\phi_{\mathrm{i}}|_1 = n_i$, $\mathrm{i} \in (1, 2, \cdots, n)$. If $(k-M)/N \to 0$, then $\mathrm{E}(\|\Phi x\|_2^2) = \|x\|_2^2$.

Proof: With the notation presented above, we are given a random vector $X(x_1, x_2, \cdots, x_n)$ and assume that its measurement can change to be (3.12):

$$\mathrm{E}(\|Y\|_2^2) = \mathrm{E}(\textstyle\sum_{i=1}^{n}|\Phi X|^2) = \mathrm{E}\left(\sum_{i,j} x_i x_j \phi_{t_i}\phi_{t_j}\right) \tag{3.12}$$

where t is all of the possible permutations of $\{1, 2, \cdots, N\}$. The index couple (t_i, t_j) ranges uniformly over all of the possible values $(1, N)$ because of the assumption, i.e., the vector X is random and every item is uniformly distributed in X. Assume one item in the i^{th} cluster is x_{l_i}. So (3.12) can change to be (3.13):

$$\begin{aligned}\mathrm{E}(\|Y\|_2^2) &= \sum_{i=1}^{n} |n_i x_i|^2 + \mathrm{E}(\sum_{i\neq j}{}_{i,j}\ x_i x_j \phi_{t_i}\phi_{t_j}) \\ &= \sum_{i=1}^{n} |n_i x_i|^2 + 1/N \sum_{l_i,l_j=1}^{M} (n_i^2 l_i^2 - n_i l_i) x_{l_i}^2 \\ &= \sum_{i=1}^{n} |n_i x_i|^2 + 1/N \sum_{i=1}^{M} (n_i^2 l_i - n_i) l_i x_{l_i}^2\end{aligned} \tag{3.13}$$

If the sensing matrix is an UMDC matrix, in other words, $n_i = 1$, $i \in (1, n)$, thus, (3.13) can change to be (3.14):

$$\mathrm{E}(\|Y\|_2^2) = \mathrm{E}(\|\Phi X\|_2^2) = \|x\|_2^2 + 1/N \sum_{i=1}^{M} (n_i^2 l_i - n_i) l_i x_{l_i}^2 \tag{3.14}$$

If we want to obtain $\mathrm{E}(\|\Phi x\|_2^2) = \|x\|_2^2$, the second item $(1/N \sum_{i=1}^{M} (n_i^2 l_i - n_i) l_i x_{l_i}^2)$ in (3.14) should be 0. Applying the Cauchy-Schwarz inequality, we can obtain (3.15):

$$\begin{aligned}\sum_{i=1}^{M} (n_i^2 l_i - 1) l_i x_{l_i}^2 &\leq \sum_{i=1}^{M} (n_i^2 l_i - n_i) \sum_{i=1}^{M} l_i x_{l_i}^2 \\ &= (\sum_{i=1}^{M} n_i^2 l_i - n_i M) \|x\|_2^2\end{aligned} \tag{3.15}$$

(3.15) can change to be (3.16).

$$1/N \sum_{i=1}^{M} (n_i^2 l_i - n_i) l_i x_{l_i}^2 \leq 1/N \, (\sum_{i=1}^{M} n_i^2 l_i - n_i M) \|x\|_2^2 = ((k - M)/N) \, \|x\|_2^2 \tag{3.16}$$

Because $k \geq M$, if $(k - M)/N \to 0$, then $(1/N \sum_{i=1}^{M} (n_i^2 l_i - n_i) l_i x_{l_i}^2) \to 0$, which means $\mathrm{E}(\|\Phi x\|_2^2) = \|x\|_2^2$.

In the next step, we prove that $\|\Phi x\|_2^2$ converges to its expectation $\mathrm{E}(\|\Phi x\|_2^2)$.

Theorem 2 [188] [189]: (Self-Avoiding McDiarmid inequality) Let $X_1, X_2, \cdots, X_m$ be the probability space, and define X as the probability space of all distinct m-tuples, which is the subset of the product set $\chi = X_1 \times X_2 \cdots \times X_m$ given by (3.17).

$$X = \{(t_1, \cdots, t_m) \in \prod_{i=1}^{m} X_i \text{ s.th. } \forall i \neq j\text{: } t_i \neq t_j\} \tag{3.17}$$

Let $h(t_1, \cdots, t_m)$ be a function from the set X to $\mathbb{R}$ such that for any coordinate i, given $t_1, t_2 \cdots, t_{i-1}$,

$$\begin{aligned}&\Big| \sup_{u \in X_i; u \neq t_n, n=1\to \mathrm{i}} \mathrm{E}[h(t_1, \cdots, t_{i-1}, u, T_{i+1}, \cdots, T_m)] \\ &- \inf_{l \in X_i; l \neq t_n, n=1\to \mathrm{i}} \mathrm{E}[h(t_1, \cdots, t_{i-1}, l, T_{i+1}, \cdots, T_m)]\Big| \leq c_i\end{aligned} \tag{3.18}$$

where the expectations are determined by the random variables $T_{i+1}, \cdots, T_m$. (For more information, see in reference [189]). If (3.18) is satisfied, then for any positive γ, (3.19) can be obtained.

$$\Pr[|h(T_1 \cdots, T_m) - \mathrm{E}[(T_1 \cdots, T_m)]| \geq \gamma] \leq 2\exp(-2\gamma^2/\sum c_i^2) \qquad (3.19)$$

We use this theorem to prove the concentration inequality.

Lemma 6: Assume that a random vector $X(x_1, x_2, \cdots, x_k)$ is a p-dissimilar vector and that $x_i \neq 0$, Every item of X is uniformly distributed in X, and its MDC sensing matrix is $\Phi_{M\times N}$. Assume that $f(X) = \sum_{i=1}^{k} x_i \phi_{p_i}$. The cluster set of this vector is $\mathrm{Set}(C) = \{C_1, C_2, \cdots, C_m\}$ and $\mathrm{I}_{\max}(\mathrm{Set}(C)) \leq N/M$. Therefore, $\Phi_{M\times N}$ satisfies (3.20).

$$\Pr[|\|f\|^2 - \|x\|^2| \geq \beta\|x\|^2] \leq 2\mathrm{e}^{-(M\varepsilon(\beta^2))} \qquad (3.20)$$

Proof: Assume that Ω_k is the set of all k-tuple permutations $(t_1, t_2, \cdots, t_k)$, which follows the definition that all entries of k-tuples of Ω_k are distinct. The set Ω_k is finite, has a counting measure and also can be renormalized to have a total mass of 1. Ω_k is the probability space of the random vector $X(x_1, x_2, \cdots, x_k)$ with k non-zero entries. Let set $T_k \sim (t_1, t_2, \cdots, t_k)$ denote a permutation of $\{1, 2, \cdots, N\}$. Because X is random and every item distributes uniformly in X, $(t_1, t_2, \cdots, t_k)$ can be regarded as being uniformly distributed in Ω_k .

Let f: $t_k \rightarrow C^M$ be defined by $f(t_1, t_2, \cdots, t_k) = \sum_{i=1}^{k} x_i \phi_{p_i}$, and let h: $t_M \rightarrow \mathbb{R}$ by $h(t_1, t_2, \cdots, t_k) = \| f(t_1, t_2, \cdots, t_k)\|_2^2$, in other words (3.21).

$$h(t_1, t_2 \cdots t_k) = \sum_{i=1}^{k} |x_i|^2 + \sum_{\substack{i,j=1 \\ i\neq j}}^{k} x_i \overline{x_j} (\phi_{t_i})^T \overline{\phi_{t_j}} \qquad (3.21)$$

Then, (3.22) can be obtained:

$$\begin{aligned} h(t_1, \cdots, t_l, \cdots t_k) - h(t_1, \cdots, t_p, \cdots t_k) &= \{\textstyle\sum_{i with\, i\neq l}[x_l \overline{x_i}(\phi_{t_l} - \phi_{t_p})^T \overline{\phi_{t_j}}] + \sum_{i with\, i\neq l}[x_l \overline{x_i} \phi_{t_j}{}^T \overline{(\phi_{t_l} - \phi_{t_p})}]\} \\ &= \{\textstyle\sum \left[x_l \overline{x_i}(\phi_{g(t_l,t_j)} - \phi_{g(t_p,t_j)})\right] + \sum \left[x_l \overline{x_i}(\phi_{g(t_j,t_l)} - \phi_{g(t_j,t_p)})\right] \end{aligned} \qquad (3.22)$$

Where $\phi_{g(i,j)}(x) = \phi_i{}^T \phi_j$.

Assume that $t_1, \cdots, t_l, \cdots, t_N$ and t_p are all different and $|\phi_i(x)|_{i=1,2\cdots,N}^2 \leq M^{-2\eta}$, $\eta \geq 0$.

From the definition of the MDC matrix, it can be obtained that $\max\left(\phi_{g(i,j)}(x)\right) = \mathrm{I}_{\max}(\mathrm{Set}(C))$ and $\min\left(\phi_{g(i,j)}(x)\right) = 0$; therefore, (3.23) can be obtained.

$$\begin{aligned}|h(t_{1,}\cdots,t_l,\cdots t_k)-h(t_{1,}\cdots,t_p,\cdots t_k)| &\le 2|x_l|\sum|x_j||\phi_{g(l,j)}(x)-\phi_{g(p,j)}(x)|\\ &\le 2|x_l|\sum|x_j|\, I_{max}(Set(C))\,M^{-\eta}\\ &\le 2NM^{-(\eta+1)}|x_l|\sum\nolimits_{j\,and\,j\neq l}|x_j|\end{aligned} \tag{3.23}$$

Assume that x_{t_l} is the largest item and has the most identical points in X ; at the same time, x_{t_p} has no identical items in X, which implies the sufficient condition of the Self-Avoiding McDiarmid inequality. Therefore, we can use this inequality to obtain (3.24),

$$\Pr[|h-E(h)|\ge\beta\|x\|_2^2]\le 2\exp(-\frac{\beta^2\|x\|_2^4M^{2(\eta+1)}}{2N^2\sum_{l=1}^{k}|x_l|^2[\sum_{j\,and\,j\neq l}|x_j|]^2}) \tag{3.24}$$

Because

$$\sum\nolimits_{l=1}^{k}|x_l|^2\left[\sum\nolimits_{j\,and\,j\neq l}\left|x_j\right|\right]^2 \;\le\; \sum\nolimits_{l=1}^{k}|x_l|^2\left[\sum\nolimits_{j=1}^{k}\left|x_j\right|\right]^2 \;\le\; \|x\|_2^2\times k\times\|x\|_2^2 = \; k\|x\|_2^4 \tag{3.25}$$

Can be combined with (3.25), (3.24) can change to be (3.26).

$$\Pr[|\|f\|_2^2-E(\|f\|_2^2)|\ge\beta\|x\|_2^2]\le 2\exp(-\frac{\beta^2M^{2(\eta+1)}}{2N^2k}) \tag{3.26}$$

Now, the statement that $\|\Phi x\|_2^2$ converges to $\mathrm{E}(\|\Phi x\|_2^2)$ is proved.

From Lemma 5, we can obtain (3.27):

$$\begin{aligned}\Pr[|\|f\|_2^2-\|x\|_2^2|\ge\beta\|x\|_2^2] &= \Pr[|h-E(h)|\ge\beta\|x\|_2^2]\\ &\le 2\exp(-\frac{\beta^2M^{2(\eta+1)}}{2N^2k})\;\le 2\exp(-\frac{\beta^2M^{2\eta+1}}{2N^2k})\end{aligned} \tag{3.27}$$

When $\eta\;=\;0$, Φ is the UMDC matrix.

Given the determined values for N and k, let $\varepsilon(x)=x/2N^2k$, then (3.27) can change to be (3.28).

$$\Pr[|\|f\|_2^2-\|x\|_2^2|\ge\beta\|x\|_2^2]\;\le 2\mathrm{e}^{-(M\varepsilon(\beta^2))} \tag{3.28}$$

Now, we prove that the UMDC matrix obeys the concentration inequality.

As mentioned above, when a matrix obeys Lemmas 5 and 6, it will satisfy the RIP, in other words, (3.29):

$$(1-\;\varepsilon)\|x\|_2^2\le\|\Phi_{UMDC}x\|_2^2\le(1+\varepsilon)\|x\|_2^2 \tag{3.29}$$

with a probability of at least $1-2\exp(-c_0(\varepsilon/2)M)(12/\varepsilon)^k,\;\varepsilon\in(0,1)$.

After proving the RIP of the UMDC matrix, a small point that we need to mention here is that the MDC matrix is based on the signal having identical points, but not all of the signals are sparse, and not all of the signals have the identical points; thus, the signal approximation is needed. In the following sections, it can be observed that for a neural signal, its approximation vector contains large numbers of identical points that can be compressed, which makes the neural signals largely compressed.

3.3 Actual Data and Methods

All of the algorithms, methods and data analysis procedures were implemented in MATLAB (Mathworks, Natick, MA).

The first dataset is obtained from an adult male rhesus macaque monkey in the Cognitive Neurophysiology Laboratory of McGill University. The data are from a recording system that contains 32 extracellular channels with a Utath 10 × 10 microelectrode array implemented in the prefrontal cortex. The data comprise three different recordings over three trials. The duration of each trial is 300s. First, data were filtered with a third-order bandpass Butterworth analog filter that utilized cutoff frequencies of 300 Hz and 7 kHz. Then, the filtered data were amplified with a gain of 80 db amplification, sampled at 30 kHz and digitized (10 bits per sample).

The second set of data was recorded from the visual cortex of a rat at the Center for Studies in Behavioral Neurobiology of Concordia University. The researchers used a stainless-steel-tipped microelectrode that had a shank diameter of 75 μm to record the data. The data were filtered with a fourth-order bandpass Butterworth analog filter, and the cutoff frequencies were between 150 Hz and 10 kHz. After the filtration, the data were amplified with a gain of 100 db, sampled at 32 kHz and digitized (10 bits). The duration of the recording was 60 s.

The third dataset comes from the NeuroEngineering Lab, University of Leicester [190]. The dataset comprises the simulated extracellular signals that were recorded from a human medial temporal lobe using intracranial electrodes. The duration of the signal is ten seconds long. The data were sampled at 32 kHz, filtered between 300 and 3000 Hz and digitized (12 bits).

First, to imitate similar recording conditions, all of the datasets were refiltered with a fourth-order non-causal Butterworth high-pass digital filter with a cutoff frequency of 300 Hz and were resampled at 24 kHz.

Then, we randomly selected ten (or five) groups of test data from three datasets and ensured that the data of every set were used. To construct the MDC matrix, two different clustering methods were used for the test: one is the core data clustering that we designed, and the other one is the agglomerative hierarchical clustering. The algorithm of the core data clustering is described in Table 3.2. Because of the comparison with the sparse signal, we also used an approximation method that was based on the Manhattan distance to construct the sparse signal. The MDC matrices in all of the simulations (except for the special explanation) are all UMDC matrices. The core data clustering uses the Manhattan distance, and the agglomerative hierarchical clustering uses the Euclidean distance.

Finally, all of the algorithms used in this article are BSBL (Block sparse Bayesian Learning algorithm), BP (Basis Pursuit algorithm) and OMP (Orthogonal Matching Pursuit algorithm), MP (Matching Pursuit algorithm), IRLS (Iterative Reweighted Least Square algorithm), StOMP (Stagewise Orthogonal Matching Pursuit algorithm) and Lasso (Least Absolute Shrinkage and Selection Operator). BSBL is BSBL_BO (groupStatLoc, learnlambda is 0, prune_gamma is -1, max_iters is 20, see [191]). BP, OMP, MP, IRLS and StOMP are from [192] using the default values. Lasso is from [193] using the default values.

3.4 Results and Discussion

In this section, several properties of the UMDC matrix are researched. First, the compression rate of a neural signal is considered. A comparison between sparsity and similarity in a neural signal with the Euclidean or Manhattan distance is given. Then, the property of the UMDC matrix is researched, which includes the RIP, the influence of the length of the signal, the difference between the NMDC and UMDC sensing matrices and the influence of the sampling rate. Moreover, the signal reconstruction under different sensing matrices, reconstruction algorithms, and other conditions is researched. Finally, a comparison between our work and the work from other literature is given; also, a period of the neural signal and its reconstructed signals is illustrated.

3.4.1 Compression Rate of the Neural Signal

The traditional compressed sensing theory is based on the sparsity of a signal, which is an approach that has limitations. Not all of the signals are sparse; thus, the change of the basis of a

signal and the approximation are two common methods for signal compression when one wants to use the compressed sensing technique. However, the change of the basis is complicated, which is not a good choice for the low-power device design; at the same time, the approximation still cannot compress large amounts of data, and neural signals in the time domain are an example.

The neural signal in the time domain is not sparse, and using an approximation still cannot compress the majority of the points. To illustrate this problem, we use 10-group test data (each group contains ten thousand points) to perform the simulation, and the results are indicated by Figure 3.1. It can be observed in this figure that the neural signal is not a sparse signal, because when MD equals 0, the sparsity of the signal is nearly equal to the length of the signal, which means that the neural signals are not sparse. If the approximation method is not used, then it is very hard to obtain a zero point. However, even though the approximation method can be used, less than twenty percent of the points can approximate to zero, when setting the 0-MD (Manhattan distance) to 2. As mentioned above, if a compressed signal can be recovered exactly, then the sparsity of the signal must be at least half of the signal. If half of the number of point must be compressed, then according to the simulation, the 0-MD must be set to 8. This number is enormous for the simulated neural signal because of the error regarding the original signal. Therefore, using the sparsity of the neural signal to make the compression is not an optimal method.

However, the degree of similarity in a neural signal is very high. According to Figure 3.1, using either the core data clustering or the agglomerative hierarchical clustering can largely compress the signal. When the inner MD (Manhattan distance) to the core data is 0.1, all of the data can be clustered into one thousand clusters; at the same time, setting the inconsistency (Euclidean distance) to be 1 for the agglomerative hierarchical clustering, eighty percent of the points can be clustered into two thousand clusters. Moreover, because the UMDC matrix obeys the CRIP or the RIP under two prerequisites, using the UMDC matrix to compress the neural signals is a very good choice.

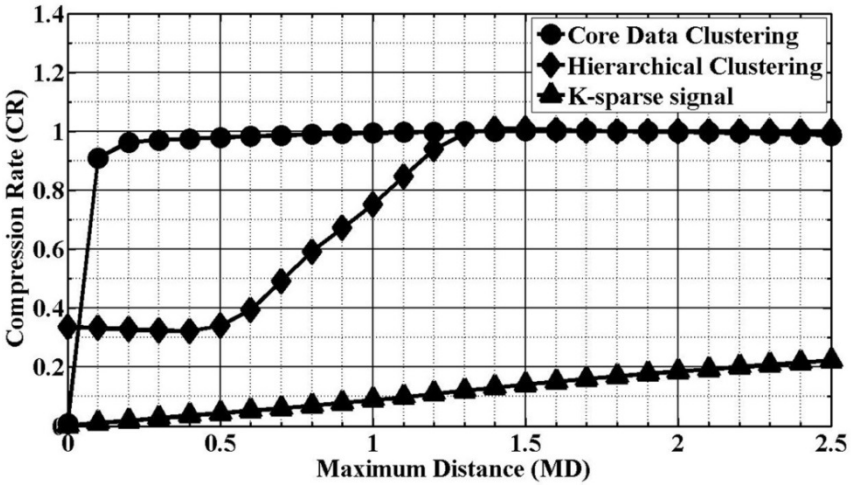


Figure 3.1 Comparison between sparsity and similarity. In the simulation, for the core data clustering method, the inner MD is the maximum Manhattan distance between each point to the core data. For the hierarchical clustering, the inner MD is the inconsistency of each cluster (point) under the Euclidean distance. For a signal, 0-MD is the Manhattan distance between a point and the zero.

3.4.2 RIP of the UMDC Matrix

In the last sub-section, the conclusion is obtained that a neural signal can be regarded as a minimum Euclidean (or Manhattan) distance *p*-dissimilar vector, which can be clustered into a small number of clusters; as a result, it can be compressed largely by the UMDC matrix. Although the CR is massive, it also needs the reconstruction error of the signal to be acceptable, which means that the UMDC matrix must obey the RIP. According to the proof in section 3.2, if the UMDC matrix obeys the RIP, there are two prerequisites: $(k-M)/N \to 0$ and $\mathrm{I}_{\max}(\mathrm{Set}(C)) \leq N/M$. Therefore, in this sub-section, both of the prerequisites are proved.

First, the relationship among K, N, and M, i.e., $\mathrm{R}(K, M, N)$, is researched. In section 3.2, if $(k-M)/N \to 0$, then $\mathrm{E}(\|\Phi x\|_2^2) = \|x\|_2^2$. We used five groups of neural signals from three datasets to build the simulation. The length of the signal in every group used for the simulation is one thousand points. We randomly selected the neural signal and repeated this process one hundred times to calculate the expectation of the measurement under different values of $\mathrm{R}(K, M, N)$. The simulation results are illustrated in Figure 3.2. In this figure, it can be seen clearly that when $\mathrm{R}(K, M, N)$ gradually decreases, the CEER also decreases without consideration

of the sparsity of the signal. When $R(K, M, N) = 0$, CEER is decreased to be zero. Additionally, when k is small (for example, $0.2N$), the UMDC matrix can compress most of the data with a small CEER, which means that if the signal is sparser, a larger CR can be applied to compress the signal. Therefore, the result that $(k - M)/N \to 0$ implies that $E(\|\Phi x\|_2^2) = \|x\|_2^2$ can be obtained.

Table 3.2 Core data clustering algorithm

A vector $X(x_1, x_2, \cdots x_n) \in \mathbb{R}^n$. Set parameter $\sigma \geq 0$;

$\{C_1, C_2, \cdots, C_n\}$, C_t is a cluster, $t \in (1, n)$; $'\Rightarrow'$ means "add into"

$L(X) = n$; $k = 1$, $i = 1$

while $k \leq n$ do

if $(x_k \notin C_d, d \in (1, i))$ do
  if $(i = 1)$
  $x_k \Rightarrow C_1$
  for $j = 1, 2, \cdots, n$ and $j \neq k$
    if $(f(\|x_k - x_j\|_2 \leq \sigma$ (or $|x_k - x_j|_1 \leq \sigma))$ and $x_j \notin C_1)$
    then $x_j \Rightarrow C_1$
  end
  $i = 1$
  else
  $x_k \Rightarrow C_{i+1}$
  for $j = 1, 2, \cdots, n$ and $j \neq k$
    if $(f(\|x_k - x_j\|_2 \leq \sigma$ (or $|x_k - x_j|_1 \leq \sigma))$
      and $x_j \notin C_d, d \in (1, i))$
    then $x_j \Rightarrow C_{i+1}$
  end
  $i = i + 1$
  else
  $k = k + 1$

end

$\{C_1, C_2, \cdots, C_i\}$ is one cluster set

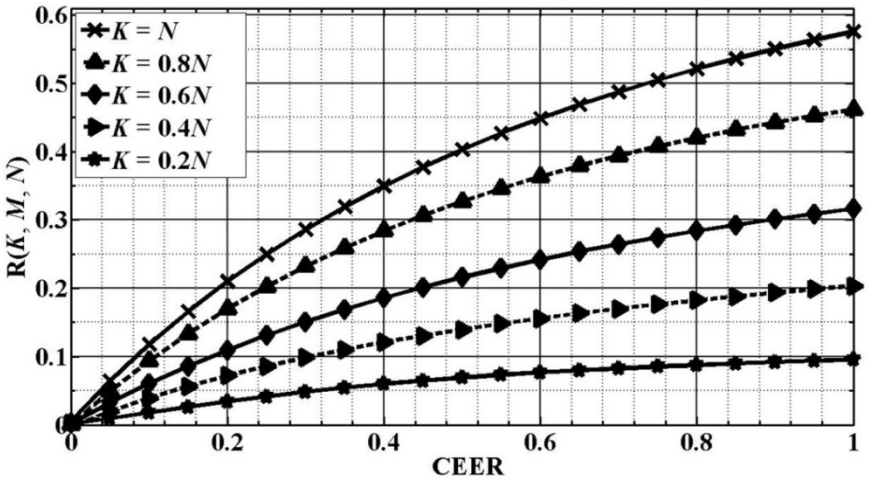


Figure 3.2 Relationship between CEER and R(*K, M, N*). The length of the data is 1000; they are randomly picked from five groups of data, and the process is repeated 100 times. CEER is the compression error of expected measurement.

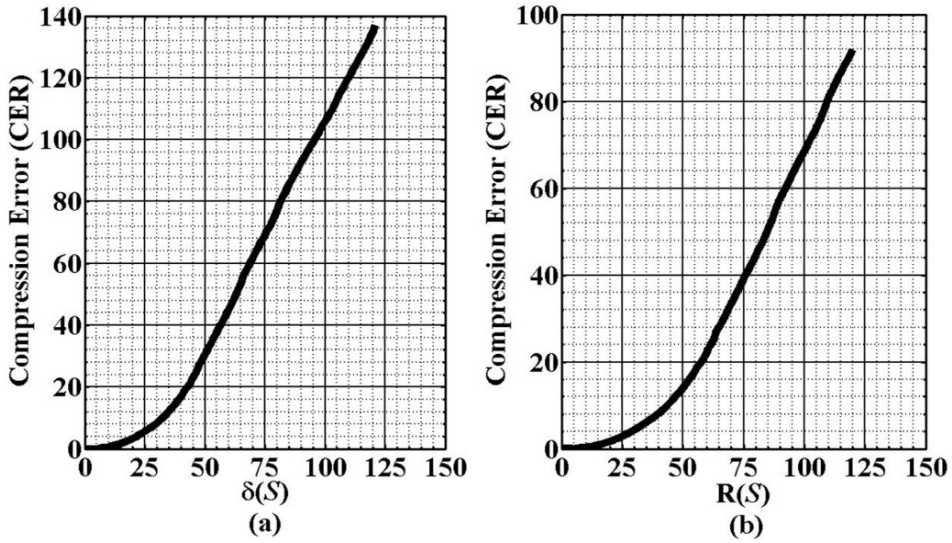


Figure 3.3 Relationship between the CER and δ(*S*) , R(*S*). Here, *N* = 1800 and *M* = 180. δ(*S*) is the Standard deviation of the size of all of the clusters in a cluster set, and R(*K*, *M*, *N*) = (*k* - *M*) / *N*

Moreover, there are more ways in which the clustering influences the CER. Two effects are researched: one is the extent of the evenness, and the other is the maximum size of the cluster in a cluster set. In the simulation, we select a period of the signal that has a length of 1800 (*N* = 1800) and compress it to 180 points (*M* = 180). First, the standard deviation δ(*S*) evaluates the evenness of the cluster. In Figure 3.3(a), it can be noted that when the δ(*S*) decreases, the CER

also reduces, which means that the cluster is clustered more evenly, and $\|\Phi x\|_2^2$ approximates deeply to $\|x\|_2^2$. In addition, it can be learned from Figure 3.3(b) that if the value of R(S) becomes smaller, the CER becomes lower, which proves that when $I_{max}(\text{Set}(C)) \leq N/M$, $\|\Phi x\|_2^2 \to \|x\|_2^2$. Therefore, if the rows of the UMDC matrix are more even, and the number of the non-zero items in the row containing the most non-zero items in the MDC matrix is smaller than N/M, then the UMDC matrix obeys the RIP.

3.4.3 Research on the Signal Reconstruction

In this sub-section, the reconstruction of the compressed signal is researched. We research the neural signal compression under different (random or deterministic) sensing matrices and when reconstructed by different reconstruction algorithms; then, we research the core data clustering and agglomerative hierarchical clustering. Finally, we research the influence of the length of the signal and compare the UMDC matrix with the NMDC matrix.

First, the UMDC matrix has the smallest RER compared with the random sub-Gaussian sensing matrix, the random DFT matrix and the LDPC (girth = 10) sensing matrix. To better compare the different sensing matrices, two groups of signals are constructed: one group is composed of the non-sparse signals, and the other group comprises the sparse signals for which D(K) = 0.5. In this simulation, we used ten groups of random signals from three datasets, and the length of every test signal is 2560. Every signal is randomly selected. Figures 3.4 - 3.6 show all of the comparisons in which there were five sensing matrices: UMDC, random Bernoulli, random discrete Fourier transform, random Gaussian and LDPC sensing matrixes. In addition, three different reconstruction algorithms are used: BSBL, BP and OMP. For the non-sparse signals, Figures 3.4(a), 3.5(a) and 3.6(a) show that the UMDC matrix has the smallest RER, although the difference is small. Additionally, with the BP algorithm, the RER of the UMDC matrix approximates zero when the CR is ninety percent, which is enormously better than the other sensing matrices with the BP algorithm. In addition, with the OMP algorithm, the RER of the UMDC matrix is obviously superb compared with the other sensing matrices. For the sparse signals, Figures 3.4(b), 3.5(b) and 3.6(b) show that the advantage of using the UMDC matrix is not obvious with the BSBL algorithm, which is nearly same with the random Bernoulli sensing matrix. However, the BP algorithm can excellently reconstruct the compressed signal. Additionally, when the CR is less than eighty percent, it has the smallest RER when using the

OMP reconstruction algorithm. Therefore, the UMDC matrix can be reconstructed by the BP algorithm exactly with a trivial RER when the CR is less than ninety percent, regardless of whether the signal is sparse or not; also, the UMDC sensing matrix has the best reconstruction performance with the BSBL or the OMP reconstruction algorithm when the simulated neural signal is non-sparse or low-sparse. In summary, using the UMDC matrix and the BP reconstruction algorithm to compress and reconstruct the neural signal is one of the optimal choices.

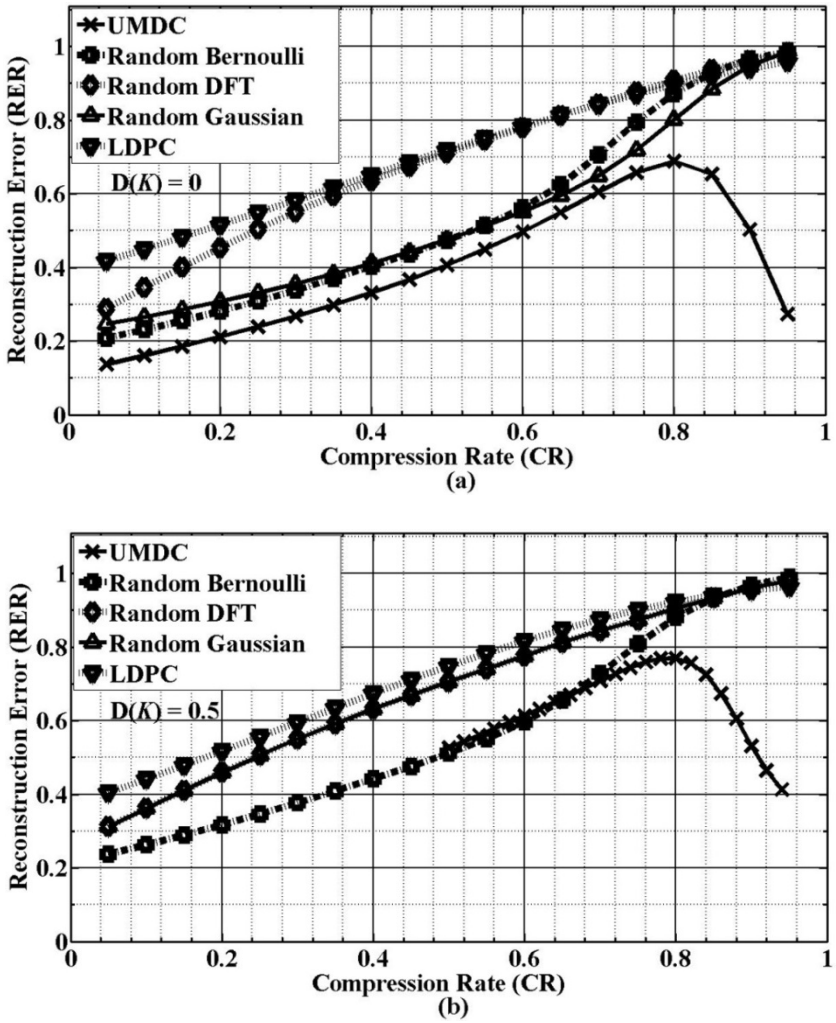


Figure 3.4 Signal reconstruction comparison with the BSBL algorithm: (a) D(*K*) = 0, (b) D(*K*) = 0.5

Moreover, two clustering methods are researched. Seven reconstruction algorithms are used for the comparison, namely, BSBL, BP, OMP, MP, IRLS, StOMP and Lasso. The test data are from ten groups of random signals. In Figure 3.7, with the same CR, the RER of the UMDC matrix

based on agglomerative hierarchical clustering is larger than the UMDC matrix based on the core data clustering, but the difference of the RER is small, which means that the UMDC matrices based on both clustering methods have a similar RER. Additionally, the complexity of the two methods is compared. We compare the algorithms with respect to their running times and memory space. The running times of the two algorithms are 1.3 ms and 23.4 ms, respectively. The occupation of memory space for the two algorithms is 468 kB and 1.5 MB, respectively. Therefore, it can be observed that agglomerative hierarchical clustering is more complicated than the core data clustering method. In addition, it can be observed that the BP and Lasso algorithms can excellently reconstruct the compressed signals, which shows that both methods can be chosen to reconstruct the compressed signals.

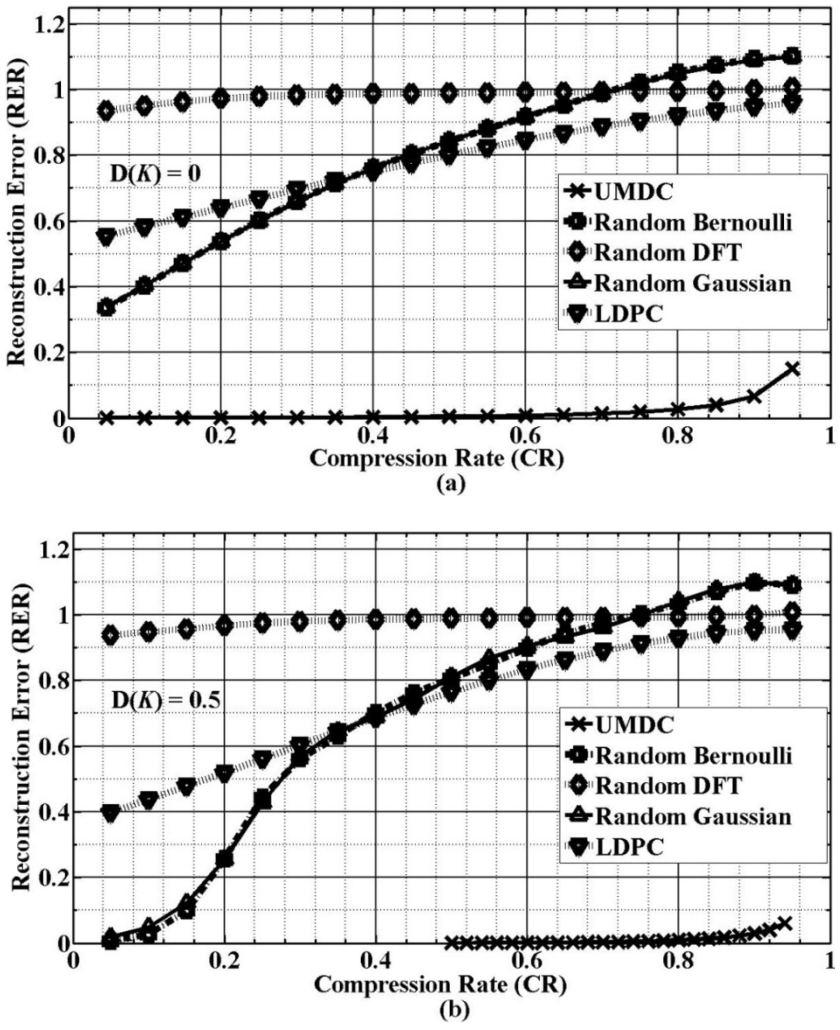


Figure 3.5 Signal reconstruction comparison with the BP algorithm: (a) D(K) = 0, (b) D(K) = 0.5

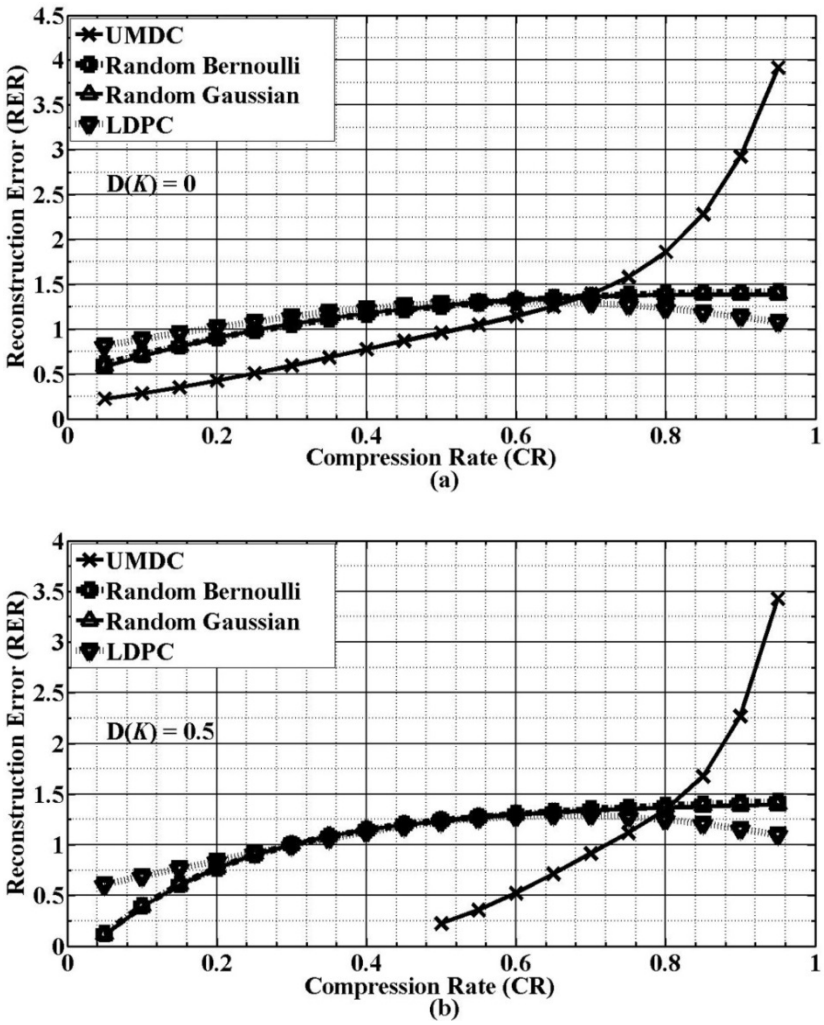


Figure 3.6 Signal reconstruction comparison with the OMP algorithm: (a) D(*K*) = 0, (b) D(*K*) = 0.5

Third, the UMDC matrix is excellent for a long-length neural signal. Five groups of random neural signals from three datasets with different lengths are researched. The *N* in Figure 3.8 is 50; thus, the lengths are 50, 500, 1000, 2500 and 5000 points. Figure 3.8 shows that under massive compression, e.g., CR = 90%, when the signal length is 50, the RER is large; however, when the data length exceeds 500, the RER is negligible (specifically, less than 0.1). In addition, under the same RER, the longer the signal is, the higher the compression rate, which means that the UMDC matrix is very suitable for long-length signal compression. For example, in Figure 3.8, under the same RER of 0.1, a 1000-point signal can be compressed to be 20 points, and a 5000-point signal can be compressed to be 50 points. Therefore, the UMDC matrix can largely compress a long-length neural signal.

Fourth, the NMDC matrix and UMDC matrix are compared. Ten groups of random neural signals are used for the simulation. Although all of the simulations are based on the UMDC matrix, it is still better to research the difference between the UMDC and NMDC matrices. Figure 3.9 shows clearly that with the BP reconstruction algorithm, the difference in the RERs of the two MDC matrices is not recognizable, which means that through the BP algorithm, using the UMDC matrix or the NMDC matrix has the same results. This finding occurs because the UMDC matrix contains only 0's and 1's, which is very useful for the hardware design. Therefore, the unit MDC matrix is the first choice for an electrical device design.

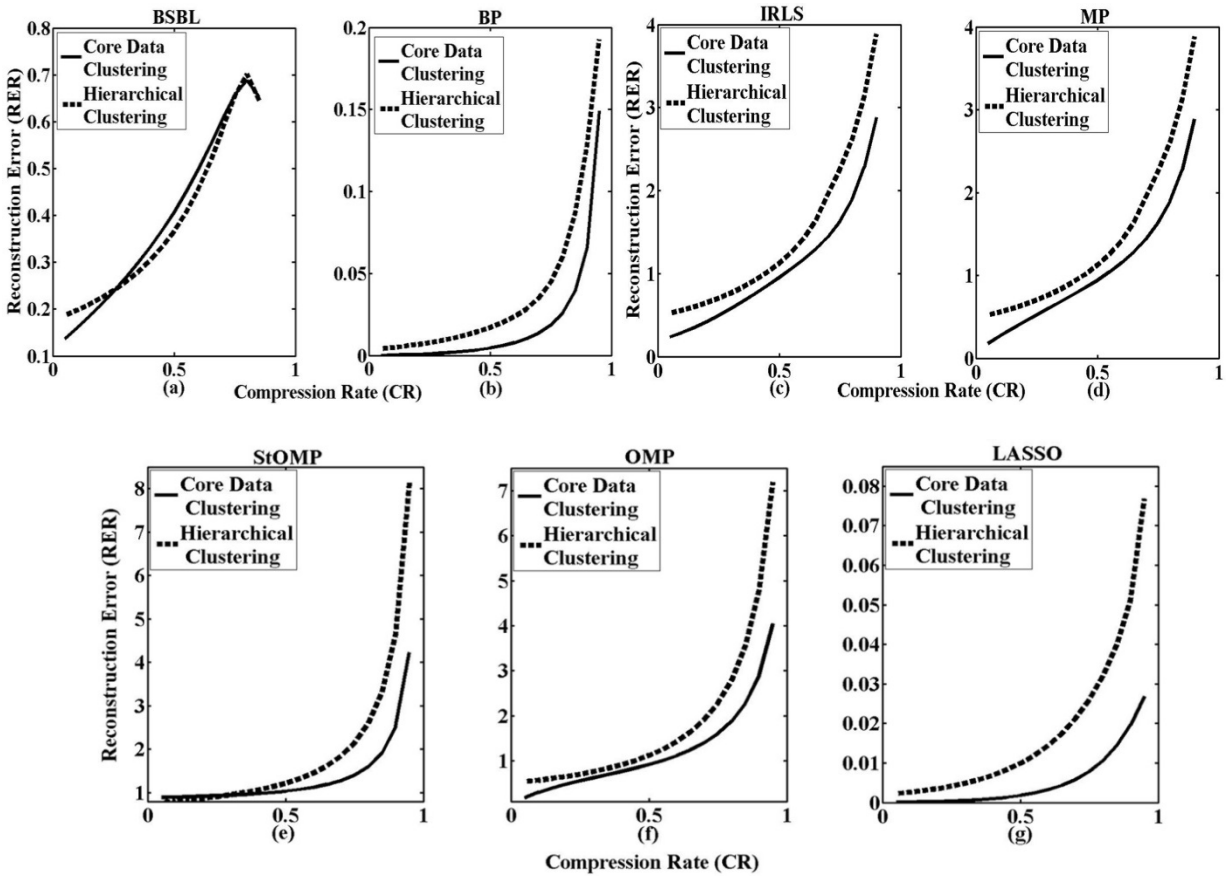


Figure 3.7 Reconstruction comparison between core data clustering and agglomerative hierarchical clustering with different reconstruction algorithms: (a) block bayesian learning algorithm, (b) basis pursuit algorithm, (c) iterative reweighted least square algorithm, (d) matching pursuit algorithm, (e) iterative threshold-selective projection algorithm, (f) orthogonal matching pursuit algorithm, (g) least absolute shrinkage and selection operator algorithm

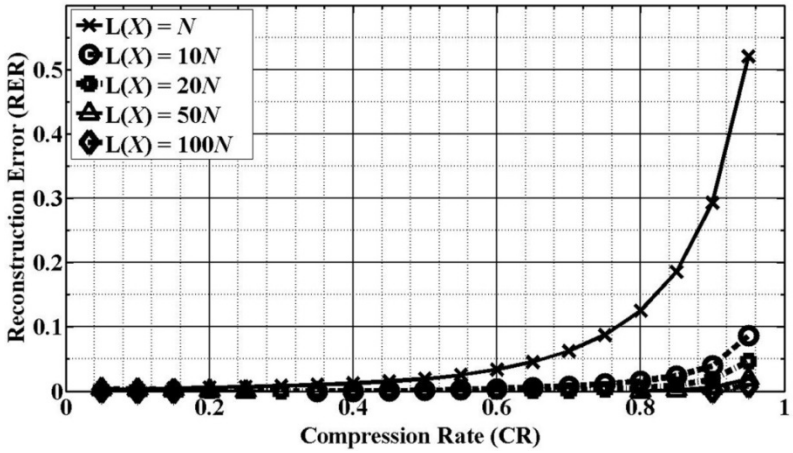


Figure 3.8 Comparison among the data with different length, N = 50

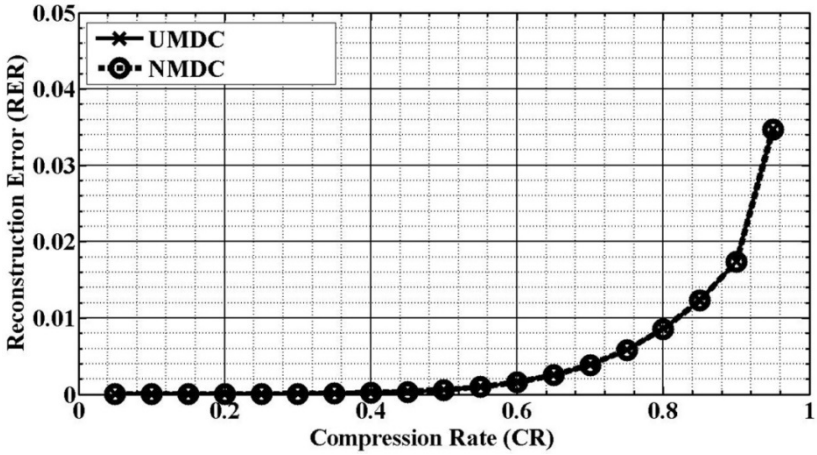


Figure 3.9 Comparison between normalized MDC and unit MDC matrices

Finally, the sampling rate does not influence the reconstruction error. A comparison of the sampling rate, compression rate and reconstruction error is shown in Figure 3.10. From the three parts of this figure, when the compression rate is determined, the reconstruction error does not change substantially when there is a change in the sampling rate. In Figures 3.10(a)-(c), when the sampling rate changes from 100 Hz to 25 kHz and the compression rate is determined, the reconstruction error does not change when the compression rate is small. Additionally, when the compression error increases, the RER fluctuates slightly, and the largest difference in the RER is approximately 0.02, 0.1 and 1, respectively. This finding indicates that under the same compression rate, the reconstruction error is not influenced by the sampling rate, and also that the MDC matrix can compress neural signals at a sampling rate of 100 Hz – 25 kHz.

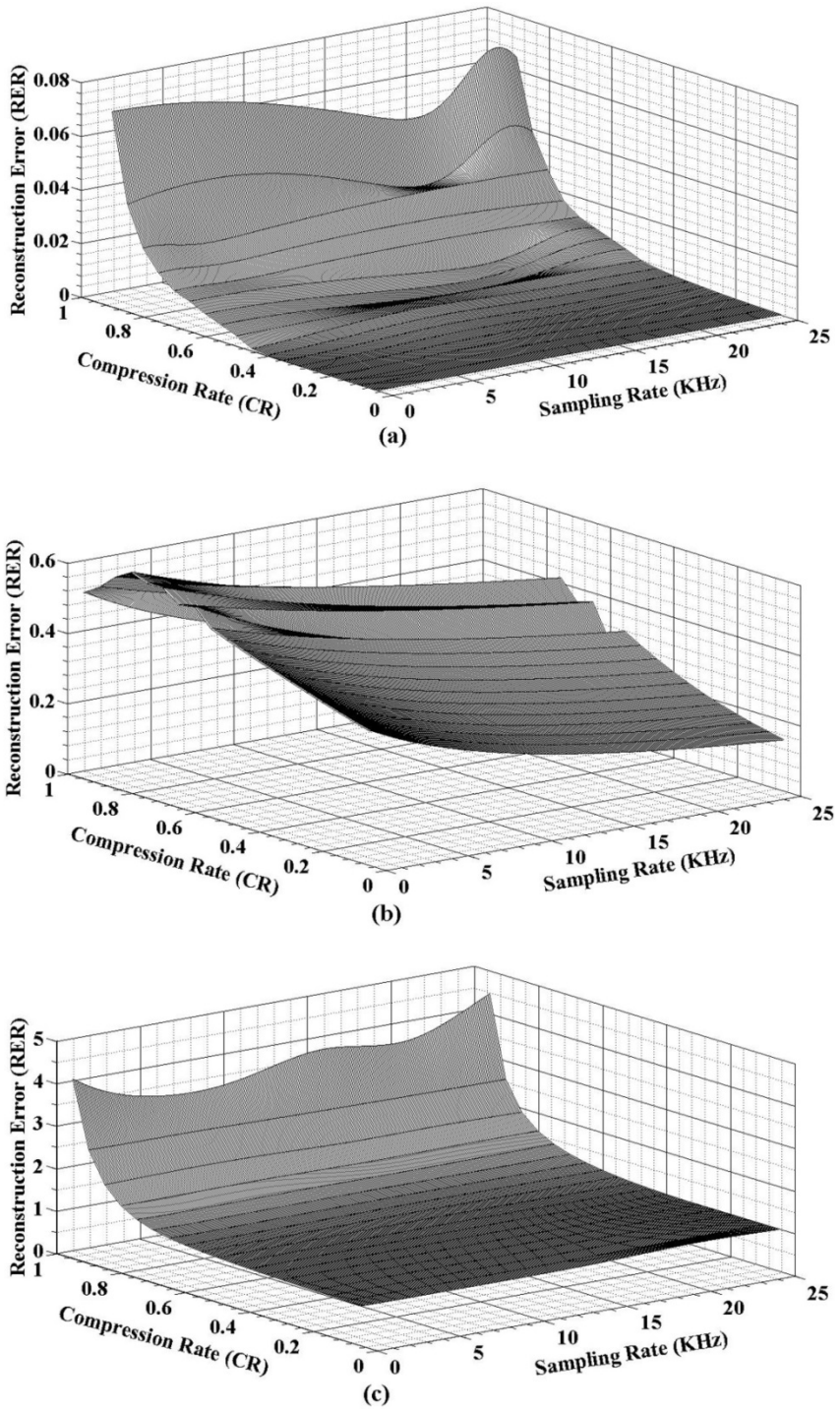


Figure 3.10 Comparison of the reconstruction results among sampling rate, compression rate and reconstruction error under three reconstruction algorithms: (a) BSBL algorithm, (b) BP algorithm, (c) OMP algorithm

3.4.4 Other Comparisons

In this sub-section, we provide a comparison between the MDC matrix and the other matrices. Then, the results between a 600-point real neural signal and its reconstructions will be shown in Figure 3.11.

Firstly, a comparison between the MDC matrix and the sensing matrices designed by other researchers is provided in Table 3.3. The comparison shows that the compared sensing matrices can compress only a highly sparse signal and that the compression rate depends strongly on the number of zero points (usually D(K) is larger than ninety percent), which cannot compress a non-sparse or low-sparse signal at a large compression rate. Nevertheless, the MDC matrix can compress a non-sparse or low-sparse neural signal with a very high compression rate; also, it has a very small reconstruction error and can reconstruct the original signal completely. Therefore, the MDC matrix has an advantage in the compression of non-sparse signals that contain identical points.

Finally, a comparison between a 600-point neural signal and its reconstructions using the BP algorithm is given in Figure 3.11. Part (a) of this figure is a 600 non-sparse neural signal. Figures 3.11(b)-(e) shows its reconstructions when CR is 90%, 96%, 98%, and 99%, and the RERs are 0.03, 0.1, 0.2 and 0.4, respectively. From Figure 3.11, it can be noted that when the CR is less than 96%, the reconstructed signal keeps a large number of details that appear in the original signal. Additionally, even if the CR is approximately 99%, the spike of the original signal is still retained very well. Therefore, the MDC matrix can be regarded as one of the optimal choices for neural signal compression.

3.5 Conclusions

In this article, first, several concepts regarding the construction of the MDC sensing matrix in a signal are presented. In addition, the construction method of the MDC matrix is given. To prove the RIP of the UMDC matrix, two prerequisites must be satisfied: the first prerequisite is that $(k-M)/N \to 0$, and the second prerequisite is that the clustering must be more even and $\mathrm{I}_{\max}(\mathrm{Set}(C)) \leq N/M$. When both prerequisites are met, we prove that given a p-dissimilar vector, the expectation of the measurement equals its ℓ_2 norm. Then the concentration inequality

and the Self-Avoiding McDiarmid inequality are applied to prove the convergence of the expectation of its measurement.

Table 3.3 Comparison between the MDC matrix and the other matrices

Ref.	**Sensing matrix**	**Data length (points)**	**CR (%)**	**D(*K*)**	**Rec.[a]Or RER**
[8]	DWT[a]	512	90	N/A	0.2
[163]	Chirp sensing codes	1681	98	0.90	N[a]
[164]	BCH	512	88	0.94	N
[164]	Ternary	2744	98	0.99	N
[141]	Elliptic curve	512	93	0.97	N
[185]	AC[a]	6561	98	0.99	N
[165]	FB[a]	6400	96	0.99	N
This work	MDC	5000	98	0	< 0.1

[a] DWT is the digital wavelet transform-based sensing matrix. FB is the Fourier-based transform sensing matrix. AC is the addictive character sequences sensing matrix. Rec. is the reconstruction. N means that it cannot perfectly recover the original signal

Moreover, five different random or deterministic sensing matrices under different reconstruction algorithms are given to prove the performance of the compression of the neural signals. To construct the MDC matrix, we use two clustering methods to construct the MDC matrix: the core data clustering and the agglomerative hierarchical clustering methods. Throughout the simulation, the MDC matrix can largely compress a neural signal, and with the BP or Lasso algorithm, the results of the reconstruction are satisfactory. Additionally, the agglomerative hierarchical clustering method is more complicated than the core data clustering method; thus, the core data clustering method is more suitable for hardware design. Second, for an MDC matrix, the longer the signal is, the larger the compression rate that can be employed under the same reconstruction error. In addition, it is proven that the UMDC matrix has reconstruction errors that are very similar to those of the NMDC matrix when using the BP reconstruction algorithm; thus, the

UMDC matrix is suitable for the hardware design of a neural recording system. Finally, the sampling rate has a slight influence on the reconstruction error.

In the end, the MDC matrix is compared with some sensing matrices from the other researchers' work. From the comparison, it can be observed that the MDC matrix has an advantage in cases that involve non-sparse or low-sparse neural signal compression. From the simulation results, we found that the RIP is too strict for the MDC matrix and that it still has some "loose" limitations for the MDC matrix; as a result, in future work, we will perform more research on these limitations. Moreover, the neural signal compression device based on the MDC matrix will be considered for implementation.

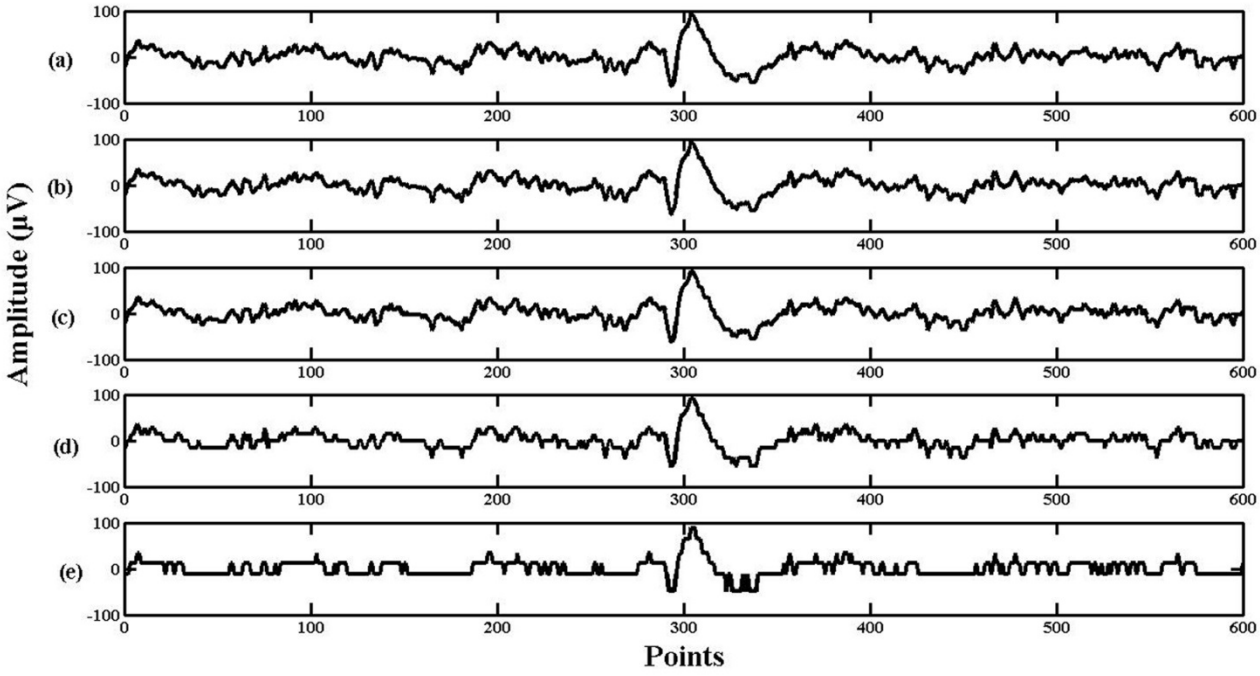


Figure 3.11 Comparison of the reconstruction results of a 600-point non-sparse neural signal using the UMDC matrix under different CRs, and the reconstruction algorithm is the basis pursuit algorithm: (a) original signal, (b)-(e) are reconstruction results with different CR and (b) CR = 90%, (c) CR = 96%, (d) CR = 98%, (e) CR = 99%

CHAPTER 4 ARTICLE 2 : AN EFFICIENT REAL-TIME NEURAL SPIKE DETECTION METHOD BASED ON BAYESIAN INFERENCE WITH AUTOMATIC TEMPLATES GENERATION

For neural signal processing inside a neural recording interface, signal reduction is another important signal processing method. Spike detection and sorting is a common method to reduce the quantity of recorded data. For the spike detection, there are mainly three categories of methods: amplitude-based, energy-based and template matching-based spike detection. Among these three methods: template matching-based method has a better detection accuracy for low SNR signals and also can make the spike classification., but this method is usually complicated, which is not easy for hardware application, and also its detection accuracy still can be improved. Therefore, it is necessary to research a high-efficiency automatic template matching-based spike detection system.

In this chapter, we put forward a novel Bayesian inference-based template matching (BBTM) spike detection and clustering method to detect and cluster spikes from noisy neural signals. Bayesian inference is applied to calculate the threshold to detect spikes, and the correlation and distance are used for spike classification. Additionally, when the templates are unknown, the BBTM method can automatically generate the templates for spike detection. Signals with different firing rates, signal-to-noise ratios and spike template generation methods are researched. Compared with some other popular spike detection methods, the BBTM method has the best detection accuracy. The false and true positive rates (FPR and TPR) based on the generated templates can reach 0.05 and 0.92 respectively for spike detection, and the average FPR and TPR (AFPR and ATPR) can reach 0.05 and 0.6 respectively for spike classification. Compared with found similar works, our proposed method displays significant advantages. Based on the analysis and discussion, BBTM method not only has a simple structure and low complexity, but also has high detection and classification accuracy.

(Biomedical Signal Processing and Control, submission date: February 2016)

An Efficient Real-Time Neural Spike Detection Method based on Bayesian Inference with Automatic Templates Generation

Nan Li[a], Liang Fang[b], and Mohamad Sawan[a]

[a] Polystim Neurotechnologies Lab. Electrical Engineering Dept., Polytechnique Montreal, 2900 Edouard-Monpetit, H3T 1J4, Montréal (QC), CANADA

[b] State Key Lab. of High Performance Computing National University of Defense Technology Changsha, Hunan, 410073

ABSTRACT — This paper puts forward a novel Bayesian inference-based template matching (BBTM) spike detection and clustering method to detect and cluster spikes from noisy neural signals. Bayesian inference is applied to calculate the threshold to detect spikes, and the correlation and distance are used for spike classification. Additionally, when the templates are unknown, the BBTM method can automatically generate the templates for spike detection. Signals with different firing rates, signal-to-noise ratios and spike template generation methods are researched. Compared with some other popular spike detection methods, the BBTM method has the best detection accuracy. The false and true positive rates (FPR and TPR) based on the generated templates can reach 0.05 and 0.92 respectively for spike detection, and the average FPR and TPR (AFPR and ATPR) can reach 0.05 and 0.6 respectively for spike classification. Compared with found similar works, our proposed method displays significant advantages. Based on the analysis and discussion, BBTM method not only has a simple structure and low complexity, but also has high detection and classification accuracy.

Keywords — Neural spike detection and classification, spike sorting, template matching, Bayesian inference, online adaptive neural signal processing

4.1 Introduction

Neural spikes are the electrical signals that neurons generate for communication with each other, which is important in the study of neuromuscular functions in the nervous systems, where the brain performs the most complex neural interactions [194] [195]. In many neuroscientific and clinical research and applications, spike detection is usually the first step in many processes and

analyses [196] [197]. Through a multichannel recording device, neural signals can be recorded from the given region, which helps researchers to investigate the activity of the given function in the nervous system [47] [198] [199]. To implement neural spike detection, some challenges need to be overcome. Primarily, the neural spikes should be extracted from the background noise. The background noise is composed of internal and external noise sources. The internal noise comes from the electrical noise that is produced in a living body and the external noise is produced by the recording devices [200]. The noise can contaminate the recorded neural signals, which causes difficulties in detecting the real spikes; therefore, correctly detecting neural spikes from the background noise is necessary. Furthermore, recording from a single unit using one electrode, or from multiple units through arrays of extracellular electrodes, the acquired neural signals are composite spikes, known as overlapping spikes, which are generated by neurons located near an electrode, but the spikes from one neuron are usually needed for research, so the separation and classification of the recorded spikes presents another challenge [201].

In order to improve the performance of neural signal processing, spike sorting is applied on recorded neural signals, which consists of spike detection and classification. Spike classification is used to identify the neurons delivering the detected spikes. It involves alignment, feature extraction, dimensionality reduction and spike clustering [88]. Among these steps, feature extraction and clustering are the two most important tasks. There are established feature extraction techniques, such as principal components analysis [171], discrete wavelet transform [172], matched subspace detector [173], etc. In [174] , the authors present the discrete derivatives (DD) method which is described as less complicated in terms of calculation while maintaining fairly high accuracy, and it is considered for use in the general circuit design. Spike clustering is the final step to sort out detected spikes from different neurons. The K-means method is a sophisticated method for the spike clustering, but it needs to manually set k in order to determine the number of required clusters [128] [175]. Some other unsupervised clustering algorithms are also discussed, such as superparamagnetic clustering [202], mean shift clustering [203], hierarchical adaptive means clustering [204].

Furthermore, spike detection is the first and an important step in spike sorting, and its goal is to separate spikes from background noise. Successful spike detection is important for accurate spike classification. According to previous research done on neural spike detection, it can be mainly divided amongst three techniques: amplitude-based detection, energy-based detection and

template matching. The amplitude-based detection method is largely used and includes root mean square (RMS) method [205], median absolute deviation (MAD) method [92], and max-min spread (MMS) sorting method [102]. The aim of the amplitude-based detection technique is mainly to calculate the standard deviation of the signals [206]. The energy-based detection method uses the change of the energy of the signal to detect spikes. Standard smooth teager energy operator (S_STEO) [103], modified smooth teager energy operator (STEO) [90], and stationary wavelet transform (SWT) [207] are some methods corresponding to this category. Amplitude-based and energy-based methods have adequate detection accuracy, but when the signal-to-noise ratio (SNR) decreases, both methods produce poor detection results [208]. Also, both methods cannot separate overlapping spikes [209]. Therefore, some other methods are needed to meet the stringent requirements of spike detection.

Template matching is a method that not only has good detection accuracy for low SNR signals, but also can separate the overlapping spikes. The template matching method applies the necessary spike templates to make the detections. A typical template matching method is based on a matching filter, such as the likelihood ratio test (LRT) detection, or the generalized likelihood ratio tests detection (GLRTs) [111], and it is very important to find a proper threshold to make the detection. Recently, several articles discuss how to use the Bayesian inference method to implement spike detection or classification [113] [114] [210]. Also, considering detection accuracy and the limits of implantable neural recording devices, an unsupervised online adaptive detection method is needed. When using template matching to detect spikes, the designed system should be able to generate the templates by itself, so several designers have tried to design an automatic template-generation system [112] [115] [211], but these systems need to be less complex and have higher detection accuracy before they see any practical use.

The contribution of this article is that we put forward a new Bayesian inference-based automatic template matching system intended to implement spike detection. We first discuss the design of a new threshold-based method using the Bayesian inference-based template matching (BBTM) method to make the detection. The BBTM method has a simple structure, which has fast calculation and is appropriate for the hardware design. Also, compared with different kinds of spike detection methods, the BBTM method has good detection accuracy. Principally, our designed system is an unsupervised adaptive online system; the system can automatically generate templates by itself and perform spike detection and classification.

Finally, the mathematical formulations and illustrations of the BBTM method are described in section 4.2. We introduce the dataset of the simulation in section 4.3. The simulation results and the corresponding discussions are given in section 4.4. In section 4.5, a conclusion is drawn.

4.2 Methods

4.2.1 Models for Spike Generation

In order to imitate the ways of real spike generation and detection, we construct and discuss the probabilistic models for spike series generation and multi-unit signal recording.

First, we construct the spike series generation model. As mentioned above, acquired signals are not usually recorded from one single neuron, that is, recorded signals are mixed signals that come from different neurons. Therefore, the spike-series generation is based on a multi-unit recording. Supposing M neurons generate M signal waveforms, we define x_t as a sampling point from one or some neuron(s) at sampling time t, that is x_t = 1, 2,….., m, where m is one neuron from M neurons. If m equals 0, then it means that the current point is noise and not from any neurons. Also, a neuron does not generate another spike within a period of time after generating a spike, which is known as the refractory period, so the probability of the generation of a spike in the refractory period is 0 for each neuron. We define the refractory period as L sampling points, and the set of neurons inside the refractory period C_{ref} can be written as $C_{ref} = \{m | \exists t' \in \mathbb{N},\ 1 \leq t' \leq L,\ x_{t-t'} = m\}$. If some neurons are not in the refractory period, and the probability of spike generation of these neurons, P_{fire}, is identical, then the set of neurons that generate spikes C_{fire} can be written as $C_{fire} = \{m | \exists t' \in \mathbb{N},\ 1 \leq t' \leq L,\ x_{t-t'} \neq m\}$. Finally, if at the sampling time t, there is no spike and the probability of the non-spike is 1- $M_{fire}P_{fire}$, where M_{fire} is the number of neurons inside C_{fire}. In summary, the model of the spike generation can be expressed as (4.1).

$$P\{x_t = m | x_{t-L}, \cdots, x_{t-1}\} = \begin{cases} 1 - M_{fire}P_{fire}, & \text{if } m = 0 \\ P_{fire}, & \text{if } m \in C_{fire} \\ 0, & \text{if } m \in C_{ref} \end{cases} \tag{4.1}$$

Second, we construct the relationship between the recorded signal and the templates. Assume that the recorded signal at the sampling time t is f_t and the template of each neuron is T_m. If the templates of all neurons have the same length, that is, L_1, then the template from each neuron can

be written as $T_m = (T_m[0], \cdots, T_m[L_1])$. Also, the background noise can be regarded as white Gaussian noise N_t with a standard deviation σ. Finally, we can use all the templates to construct a spike series which only contains single and composite spikes (S_t). Based on the assumptions, we can construct the relation among the recorded signal, spikes and the noise, that is (4.2),

$$f_t = S_t + N_t \tag{4.2}$$

Considering the noise is Gaussian white noise, the recording models can be written in (4.3) [113].

$$P(f_t \mid x_t) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(f_t - S_t)^2}{2\sigma^2}) \tag{4.3}$$

where x_t is the spike template(s) that a spike point belong to at the sampling time t, and S_t is the correspondent spike-template points. Specifically speaking, at the sampling time t, a recorded point may contain only noise point or noise and spike points. If the recorded point f_t contains spike point S_t (S_t can be from one unique template or can be composite spike point from several templates), the noise point can be acquired when the spike point is removed.

4.2.2 Bayesian Inference Analysis

After constructing the recording model, we need to further simplify this model and make it more practical for use. Considering a period of recorded data point *f'*, we want to know whether or not it is the point from a spike, and also we want to find which template the point comes from, that is, we want to find the maximal conditional probability $P(x' \mid f')$, which can be written as (4.4),

$$x'_{opt} = \arg\max_{x'} P(x' \mid f') \tag{4.4}$$

Using the Bayesian inference, we can obtain (4.5):

$$x'_{opt} = \arg\max_{x'} \frac{P(f' \mid x')P(x')}{\sum_i P(f' \mid i)P(i)} \tag{4.5}$$

where $P(f' \mid x')$ is the data likelihood when knowing the data templates. According to [114], without considering all unnecessary computation; e.g. the denominator in (4.5), (4.5) can be simplified to a new function (4.6),

$$\bar{F}(f) = P(f \mid x)P(x) \tag{4.6}$$

where $P(x)$ is the prior probability of a spike based on a template (P_{fire} in (4.1)). Combining (4.3) with (4.6), we can obtain (4.7),

$$\bar{F}(f)=\frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(f-S)^2}{2\sigma^2})P(x) \tag{4.7}$$

Because $1/\sqrt{2\pi}$ is an unchanged parameter, it can be omitted; also, we can take the logarithm, so (4.7) can be further simplified to (4.8),

$$F(f)=\ln(\bar{F}(f))=-\ln(|\sigma|)-\frac{1}{2}\frac{(f-S)*(f-S)}{\sigma}+\ln(P(x)) \tag{4.8}$$

According to [114], (4.8) can be further simplified to (4.9),

$$F(f)=(f*S-\frac{1}{2}S*S)+\ln(P(x)) \tag{4.9}$$

4.2.3 Spike Detection Based on Template Matching

The discriminant for detection is built up. Suppose a spike, S_m, occurs during $t\in(t'+1,t'+L)$ and it contains a complete template T_m from neuron m (m = 1, 2…., n). So S_m can be written as (4.10),

$$S_m=\begin{cases}T_m, & \textit{no overlap signals}\\ T_m+\lambda, & \textit{overlap signals}\end{cases} \tag{4.10}$$

If there are no overlap spikes, S_m is the only template T_m; if this spike is an overlapping spike from the templates, λ represents the signals from the other whole or parts of the templates. So (4.9) can change to (4.11),

$$\begin{aligned}F(f)&=(f^T(T_m+\lambda)-\frac{1}{2}(T_m+\lambda)^T(T_m+\lambda))\frac{1}{\sigma}+\ln(P(x_m))\\&=(f^TT_m+f^T\lambda-\frac{1}{2}T_m{}^TT_m-T_m{}^T\lambda-\frac{1}{2}\lambda^T\lambda)\frac{1}{\sigma}+\ln(P(x_m))\\&=(f^TT_m-\frac{1}{2}T_m{}^TT_m)\frac{1}{\sigma}+(f^T\lambda-\frac{1}{2}\lambda^T\lambda)\frac{1}{\sigma}-\frac{T_m{}^T\lambda}{\sigma}+\ln(P(x_m))\end{aligned} \tag{4.11}$$

where σ is the standard deviation of the signal, $P(x_m)$ is the prior probability of a spike (point) from template T_m

In (4.11), $(F(f)-\left(f^T\lambda+T_m{}^T\lambda+\frac{1}{2}\lambda^T\lambda\right)\frac{1}{\sigma})$ is a bounded value; therefore, to simplify (4.11), we suppose that there exits an α, which satisfy (4.12),

$$-\frac{\alpha}{2\sigma}\times(T_m{}^T T_m)\le F(f)-(f^T\lambda+T_m{}^T\lambda+\frac{1}{2}\lambda^T\lambda)\frac{1}{\sigma}\le\frac{\alpha}{2\sigma}\times(T_m{}^T T_m) \tag{4.12}$$

Combining (4.11) with (4.12), (4.13) can be written,

$$-\frac{\alpha}{2}\times(T_m{}^T T_m)\le f^T T_m-\frac{1}{2}(T_m{}^T T_m)+\sigma\ln(P(x_m))\le\frac{\alpha}{2}\times(T_m{}^T T_m) \tag{4.13}$$

From (4.13), we extract (4.14)

$$(1-\alpha)(T_m{}^T T_m)-\sigma\ln(P(x_m))\le f^T T_m\le(1+\alpha)(T_m{}^T T_m)-\sigma\ln(P(x_m)) \tag{4.14}$$

where α is the threshold control parameter (TCP). Assume $F'(f)=f^T T_m$, we find the discriminant to make the detection. Also, in the later section, we research and find the TCP value for the best detection performance.

Moreover, we need to research the cases that include no spikes and only noise. When a period of the signal only contains noise, $E(f^T T_m)=0$; therefore, if the signal contains a spike, then $F'(f)\neq 0$, that is, $0\le\alpha<1$. Now, we find a discriminant to detect the spike. When we do not know which template a spike comes from, we need to use this discriminant with each template to make the detection.

4.2.4 Bayesian Inference-based Template Matching (BBTM) Method

From (4.14), we acquire a threshold-based template matching system to detect the spikes. This method contains two categories: 1) detection with known templates, and 2) detection with unknown ones. The whole process of the two categories is shown in Figure 4.1. When the templates are known, two thresholds are calculated to detect a spike. The inner product of the template is calculated, then the threshold control parameter α is used to adjust the upper and lower bounds to form two thresholds. The inner product of the signal and the template are also calculated, and this product is compared with the thresholds. If the product is located between two thresholds, one spike is detected out.

When the templates are not known, the generation of templates needs to be added. Template generation can be divided into four steps: spike detection, alignment, feature extraction and spike clustering. Spike detection can mainly be achieved by the amplitude-based and energy-based detection methods, which are discussed in a later section. All spikes are aligned by the maximum

slope, and the DD method is selected to perform the feature extraction. The last step is using the K-means method to cluster the spikes and form the templates. When using the K-means method, the number of templates needs to be given. The K value can be given directly, but the BBTM method also includes an Osort algorithm to generate the number of templates *k*. The Osort algorithm is given in [112] [116].When a spike *T* is detected, the Euclidean distances between *T* and the core points of all the clusters *D* is calculated. If this distance is larger than the threshold η_1, then a new cluster is created, or else, this spike is clustered into the correspondent cluster, then we recalculate the distance of all the clusters D_c. If D_c is smaller than the threshold η_2, then two clusters are merged. To simplify the calculation, we assume $\eta_1 = \eta_2$, and they are based on the variances of the detected spikes [116]. After generating the templates, the spike detection is the same in the case of detection with known templates.

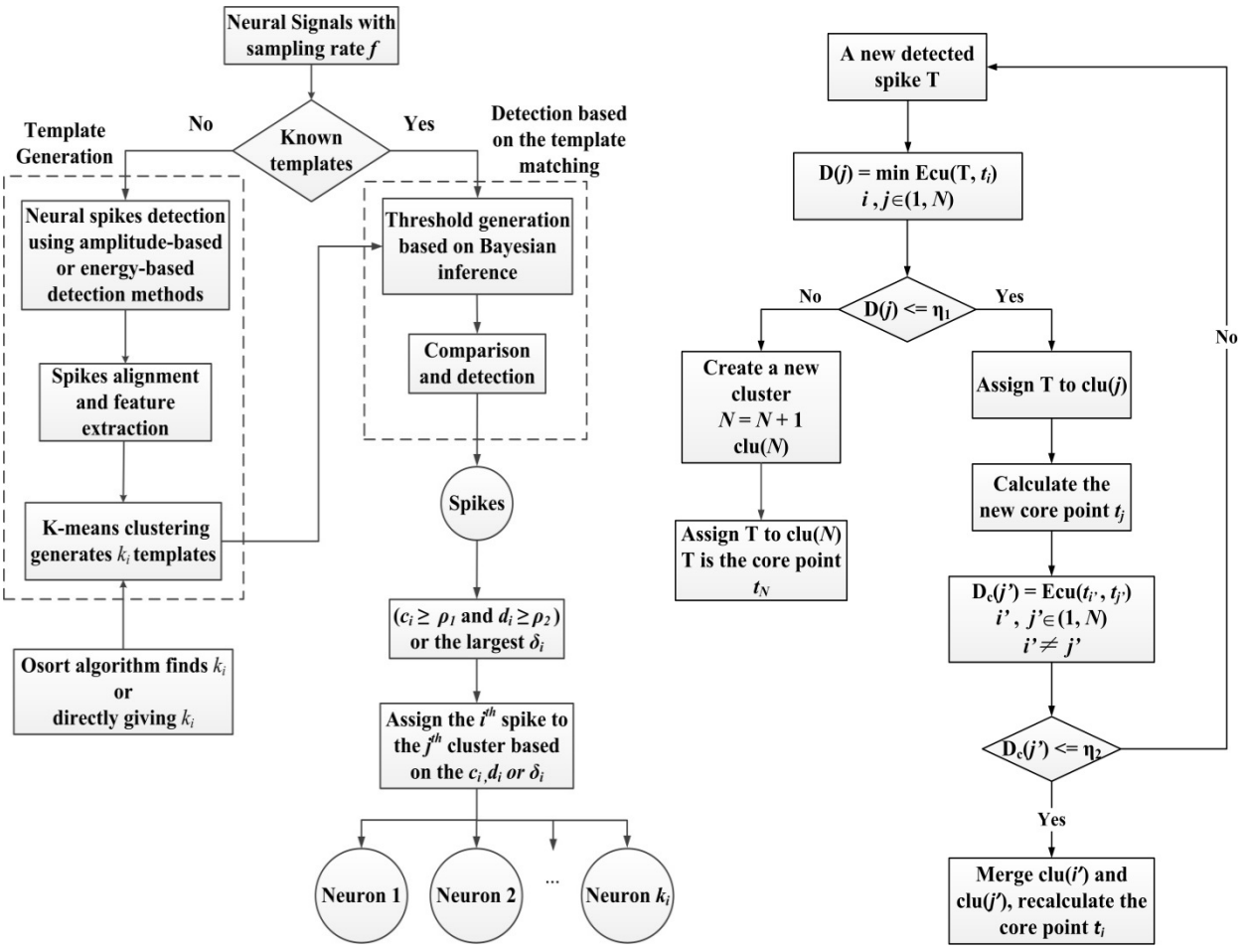


Figure 4.1 Block diagram of proposed methods (a) BBTM method (b) Osort algorithm

Beyond detecting the spikes out, it needs to know the detected spikes belong to which template, that is, the spike classification is needed. We use two factors, correlation and distance, to perform the spike clustering. Two thresholds, ρ_1 and ρ_2, are first used to cluster the spikes. Based on the experimental data, ρ_1 is chosen as 0.8 and ρ_2 is chosen as 0.5. After the first step, we use another parameter δ_i to cluster the spikes, δ_i is shown in (4.15).

$$\delta_i = a*c_i + b*(1-d_i') = a*c_i + b*d_i,\ i \in (1, M) \tag{4.15}$$

where c_i is the correlation coefficient between the detected spikes and the templates, d_i' is the normalized minimum distance between the detected spikes and the templates, a and b are weight parameters, and we use two experimental coefficients, that is, $a = b = 1$. If c_i and d_i between a detected spike and one template are bigger than ρ_1 and ρ_2, then we consider that this spike comes from that template. If c_i or d_i do not exceed the correspondent thresholds, then we classify this spike within the template that corresponds to the largest δ_i.

4.3 Test Dataset

Algorithms, methods and data analysis procedures for the proposed design were developed within MATLAB environment. The corresponding software and programs are running on a 3.4 GHz Intel I7 processor with 16 Gb of main memory.

To better evaluate the detection performance, we used synthetic neural signals that come from real recorded neural signals. With the synthetic data, we can build signals with different FR, SNRs, etc.

The neural signal dataset is acquired from the prefrontal cortex of an adult male rhesus macaque monkey (Cognitive Neurophysiology Laboratory, McGill University). The recording circuit contains 32 extracellular channels with a Utath 10×10 microelectrode array. The available dataset includes three different recordings over three trials. The duration of each trial is 300s. The data, firstly, were filtered with a third-order bandpass Butterworth analog filter with cutoff frequencies of 0.3 and 7 kHz. Then, the filtered data were amplified with a gain of 80 db amplifier, sampled at 30 kHz and digitized (10 bits per sample).

To imitate similar detection conditions, all the datasets were refiltered with a fourth-order non-causal Butterworth high-pass digital filter with a cutoff frequency of 300 Hz, resampled at 24

kHz and requantized with 10 bits per sample. Then, we extract the neural spikes. The spikes are selected from thirty-two groups of real neural signals that are described above. We first detected spikes, then we performed the feature extraction and clustering; at last, we selected three different groups (templates) of spikes to build composite signals. The length of a neural spike series is 2 ms (48 samples per spike).

After the construction of the spike series, we inserted them into the white Gaussian background noise. Signals from one neuron are first built, which is achieved by inserting one unique spike template into the background noise with a Poisson firing model using a refractory period of 2 ms, and the firing rate is set between 10 and 100 spikes per second. The prior probability $P(x)$ depends on the firing rate and can be regarded as a constant [114]. In this article, we assume that all the neurons have the same firing rates. At the sampling rate of 24 kHz, $P(x)$ are set as two constant values, 0.02 and 0.2, for the firing rates 10 and 100. To imitate composite signals from three neurons, three groups of the constructed neural signals built with three different templates are synthesized to be the final three-neuron composite neural signals. Also, we build four groups of signals with an SNR (see (4.16)) from 3 to 6 [90] [96]. This definition of SNR is a good choice for the estimation of spike detection, because it prevents errors from the relative frequency of spikes of different amplitudes [96].

$$\text{SNR} = \frac{\text{Maximum magnitude of the spike waveform}}{\text{Standard deviation of background noises}} \tag{4.16}$$

To evaluate the detection performance, we use a receiver operation characteristic (ROC) curve. The true positive rate (TPR) is the ratio between the number of the spikes that are correctly detected out (True positive, TP) and the number of spikes that are not detected (False negative, FN), and false positive rate (FPR) is the ratio between the number of times that noise is detected as spikes (False positive, FP) and the number of times that noise is correctly detected as noise (True negative, TN). Both ratios are defined in (4.17) and (4.18).

$$TPR = TP / (TP + FN) \tag{4.17}$$

$$FPR = FP / (FP + TN) \tag{4.18}$$

Also, to evaluate the classification accuracy of each neuron, we use the average true positive rate (ATPR) and the average false positive rate (AFPR). ATPR and AFPR for three neurons are

defined in (4.19) and (4.20).

$$ATPR = (TPR_1 + TPR_2 + TPR_3)/3 \tag{4.19}$$

$$AFPR = (FPR_1 + FPR_2 + FPR_3)/3 \tag{4.20}$$

4.4 Results and Discussion

In this section, we analyze the accuracy of the neural spike detection using known and unknown templates. The analysis is based on different firing rates. We select several related methods: RMS, MAD, MMS, S_STEO and STEO to make comparisons. For the case of unknown templates, we compare BBTM with other spike detection methods based on the signals with an SNR from 3 to 6. Additionally, we discuss the classification accuracy of the detected spikes and the influence of the TCP. Finally, we give results about the spike detection and classification of real neural signals.

4.4.1 Spike Detection with Known Templates

We compare the detection accuracy between BBTM and other detection methods using the signals with an SNR from 3 to 6. The firing rate of the signal is 10 in Figure 4.2. In Figure 4.2(a), it can be seen that when the FPR equals 0.05, the BBTM method has the highest TPR around 0.95, which is nearly two times larger than the amplitude-based methods. The second largest TPR is the amplitude-based method, and for the three amplitude-based methods, the detection accuracy is almost the same, which is around 0.45. The third largest TPR is among the energy-based detection methods (S_STEO and STEO), which are around 0.2. Moreover, in Figures 4.2(b)-(d), when the FPR equals 0.05, for the signals with SNR 4-6, the TPRs gradually increase, which are very close to 1 when using the BBTM method. For the three amplitude-based spike detection methods, the TPRs increase from 0.65 to 0.85 when the SNR increases from 4 to 6. For the energy-based method, we acquire similar results, and the TPRs increase with the rise of the SNR. Therefore, we can make two conclusions that for a signal with the firing rate of 10, the BBTM has the best detection accuracy under a small FPR, and also with the increase of the SNR, the detection accuracy of all the detection methods increases, but for the BBTM method, the difference in the detection accuracy is small. Finally, in our simulation for a three neuron composite signals, comparing with the dot product of the template vector (${T_m}^T T_m$), the second

item ($\sigma ln(P(x))$) of discriminant is usually very small.

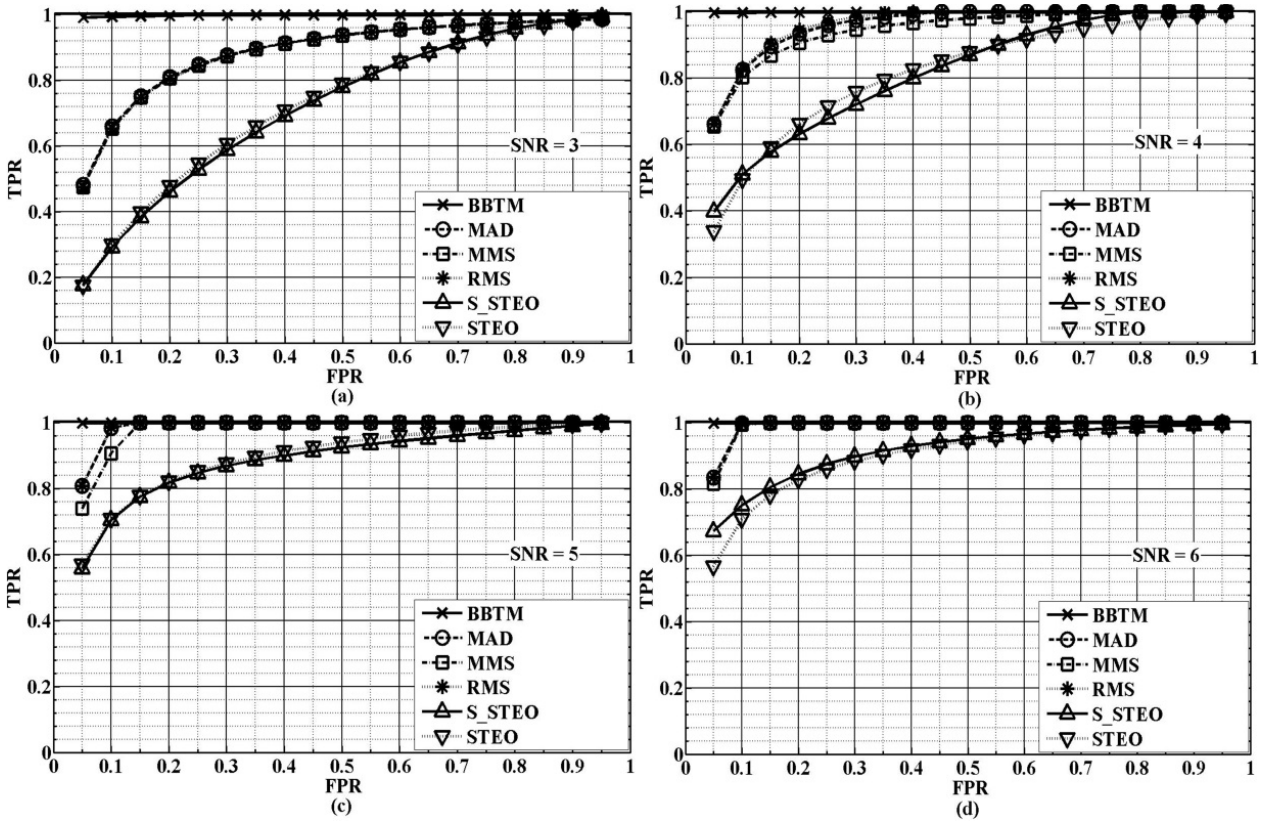


Figure 4.2 Comparison between BBTM and MAD, MMS, RMS, S_STEO and STEO methods with firing rate equaling 10, (a) SNR = 3, (b) SNR = 4, (c) SNR = 5, and (d) SNR = 6

Moreover, all the methods are compared using the signal with a firing rate of 100. First, in Figure 4.3, it can be seen that when FPR equals 0.05, for all the simulated methods, the detection accuracy is worse than the signal with firing rate 10. For the BBTM method, when the FPR and the SNR equals 0.05 and 3, the TPR is around 0.9, and the difference between the firing rate 100 and 10 is small, which can be regarded as negligible. When the SNR increases, the TPR also increases. Furthermore, when the FPR equals 0.05 and SNR of signals equals 3-6, the TPRs of the other five methods are low for both firing rate 10 and 100, which is obviously smaller than that of BBTM method. Therefore, we can conclude that, when the firing rate becomes higher, the BBTM has the best detection accuracy under a small FPR, though the detection accuracy of all methods decreases, but the BBTM method still maintains high detection accuracy, which means the BBTM method is robust.

4.4.2 Spike Detection with Unknown Templates

When the spike templates are not known, they need to be first generated. In this section, we mainly discuss how to use two kinds of spike detection methods, MMS and S_STEO methods, to perform the spike detection when the spike templates are not known, and we also use the signals with an SNR from 3 to 6, and firing rates of 10 and 100. For both MMS and S_STEO methods, there is a coefficient *P* to adjust the threshold to make the detection, which causes changing detection accuracy [96].

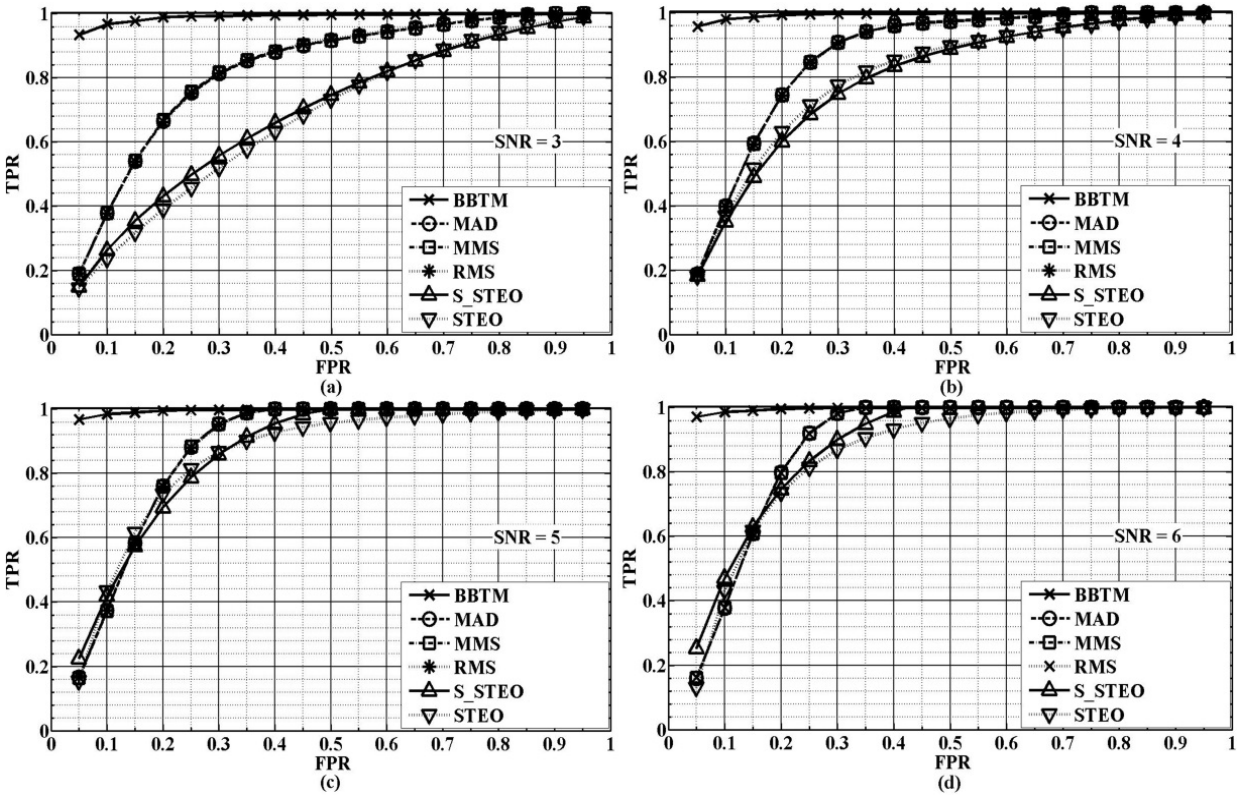


Figure 4.3 Comparison between BBTM and MAD, MMS, RMS S_STEO and STEO methods with firing rate equaling 100, (a) SNR = 3, (b) SNR = 4, (c) SNR = 5, and (d) SNR = 6

The spike detection performance using the MMS detection method is researched. When the firing rate equals 10, it can be found that choosing *P* as 4 and 5 has the biggest TPR for all the signals with an SNR from 3 to 6, which can be seen in Figure 4.4. Comparing with Figure 4.2, the TPR is slightly smaller than that of the known templates. When the FPR is 0.05, the TPR exceeds 0.9 for the signal with an SNR from 3 to 6 in Figure 4.4. Therefore, when using the MMS method for detection, choosing 4 or 5 is the best option. Also, it can be gathered that for a signal with a large

SNR, the TPR is also large. When the firing rate and the FPR equal 100 and 0.05 respectively, in Figure 4.5, if *P* equals 4, then the TPR is the largest, which is around 0.5. For the signals with SNR 4 and 5, *P*'s equaling 3 and 4 have the biggest TPR, which is around 0.6. For the signal with SNR 6, a *P* equaling 4 has the largest TPR, which is around 0.7. Comparing with Figure 4.3, it can be found that when FPR equals 0.05, the BBTM method has better detection performance. Third, considering both cases, choosing *P* as 4 or 5 is the best choice for detection when using the MMS method.

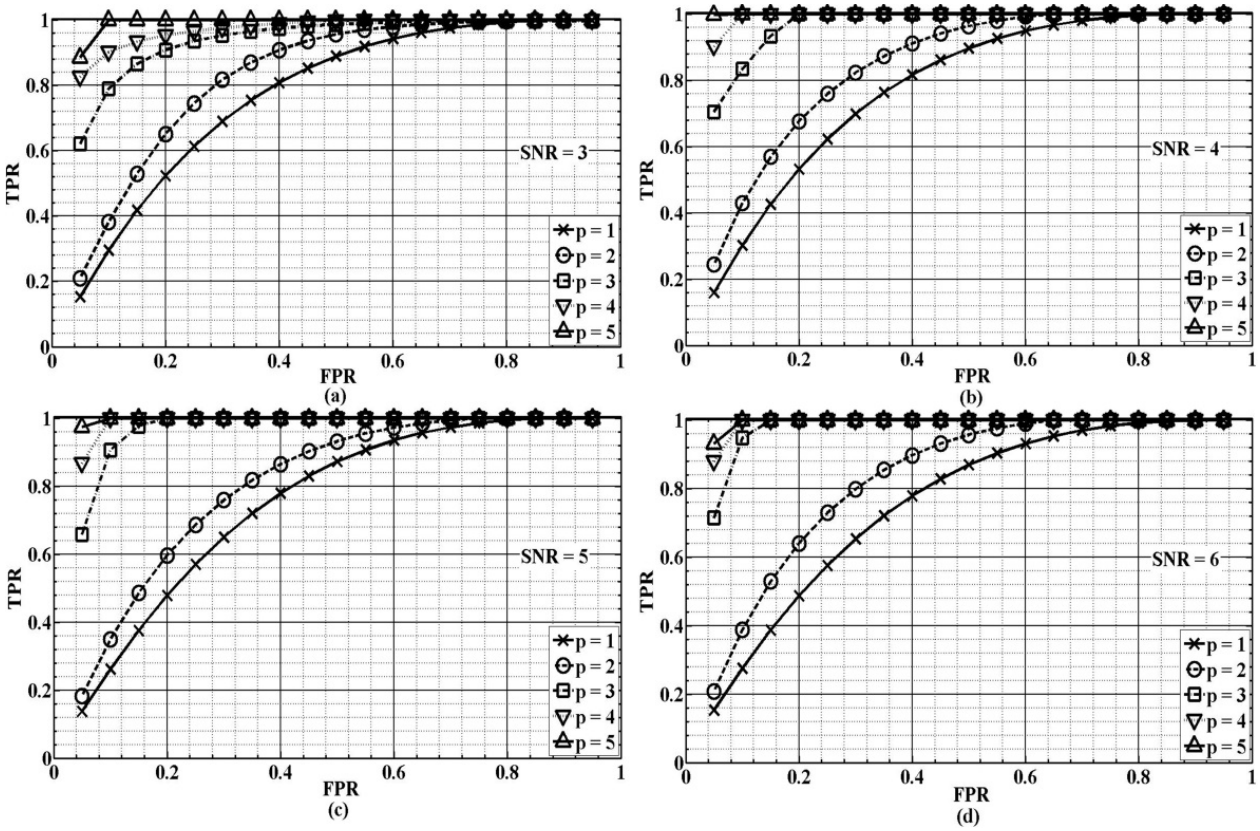


Figure 4.4 BBTM spike detection using MMS to generate spike templates with firing rate equaling 10, (a) SNR = 3, (b) SNR = 4, (c) SNR = 5, (d) SNR = 6

For a better comparison, we use the energy-based method, STEO, to generate the spike template. In Figure 4.6 and Figure 4.7, it can be found that when firing rate and FPR equal 10 and 0.05 respectively, the TPR is less than 0.2 for the signal with an SNR from 3 to 6, but when firing rate reaches 100, the TPR reaches around 0.9; especially, when *P* equals 4 and 5, the TPR can reach the highest detection accuracy, so 4 or 5 can be chosen to generate the spike templates. Considering the detection accuracy of MMS and STEO methods, it can be found that using

STEO has better detection accuracy when the firing rate is large. When the firing rate is 10, the detection accuracy largely decreases, and the difference in detection accuracy for the MMS method between firing rates 10 and 100 is smaller than that of the STEO method, so using the MMS method for detection is more robust.

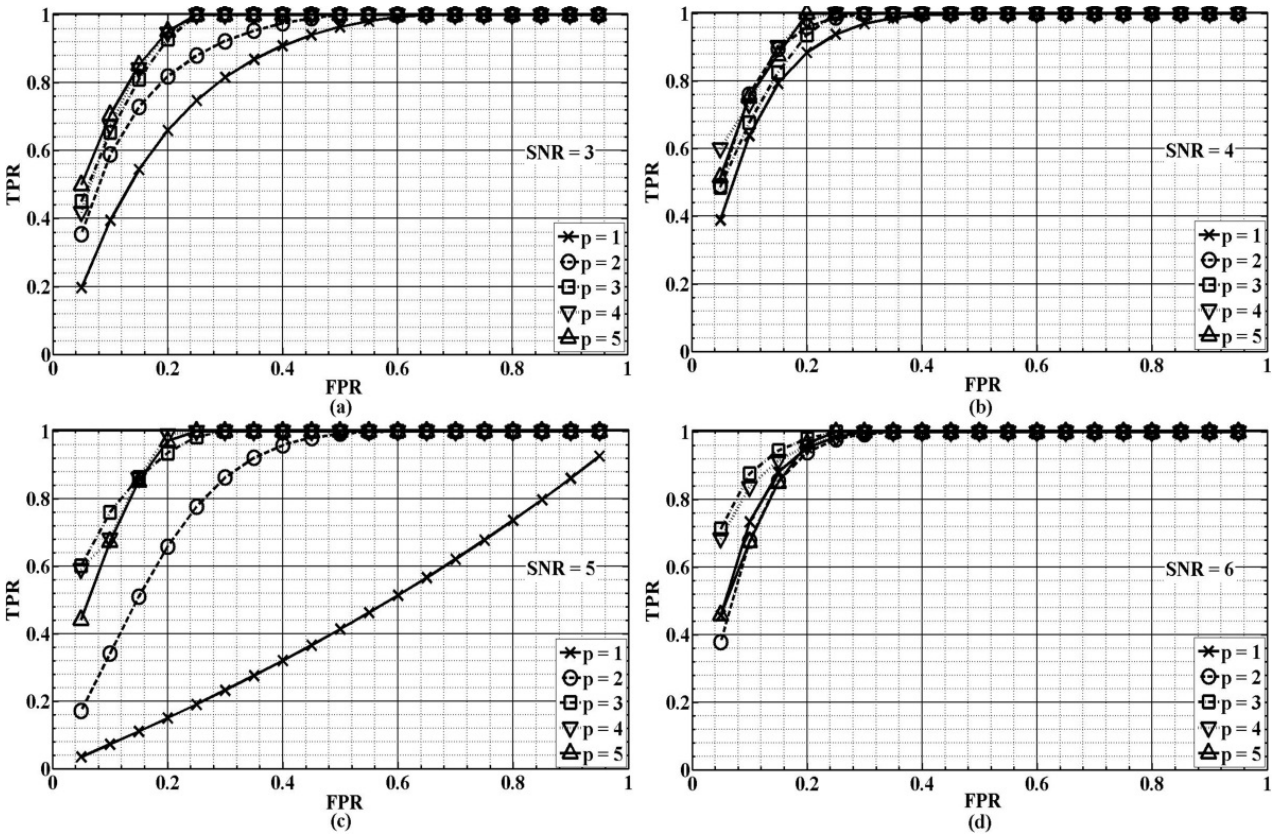


Figure 4.5 BBTM spike detection using MMS to generate spike templates when firing rate equals 100, (a) SNR = 3, (b) SNR = 4, (c) SNR = 5, and (d) SNR = 6

Moreover, using the BBTM method with the unknown templates yields better detection accuracy compared with the amplitude-based and energy-based methods. When comparing Figures 4.6 and 4.7 with Figures 4.4 and 4.5, it can be found that using the MMS detection in the BBTM method has better detection accuracy than the one which only uses the MMS detection method. When the firing rate is 10 and *P* equals 5, the detection accuracy is larger than that of the RMS, MMS and MAD methods, especially for an SNR equaling 3 and 4. Also, when firing rate is 100, the detection accuracy is 3 or 4 times larger than that of the other amplitude-based methods, which shows the BBTM method has better detection accuracy.

4.4.3 Spike Clustering and Threshold Control Parameter

The BBTM method involves spike clustering. The clustering method is described in section 4.3. Figure 4.8 shows the clustering accuracy based on the MMS and STEO methods, respectively. Also, in this figure, it can be found that the templates from the MMS method have better clustering accuracy than that of the STEO method. For the signal with SNR 3 and 4, when AFPR is 0.05 and *P* equals 4, the ATPR is more than 0.5, and for the signal with SNR 5 or 6, this rate is around 0.65. When increasing the AFPR to 0.1, the ATPR is close to or equals 1 for the signal with an SNR from 3 to 6.

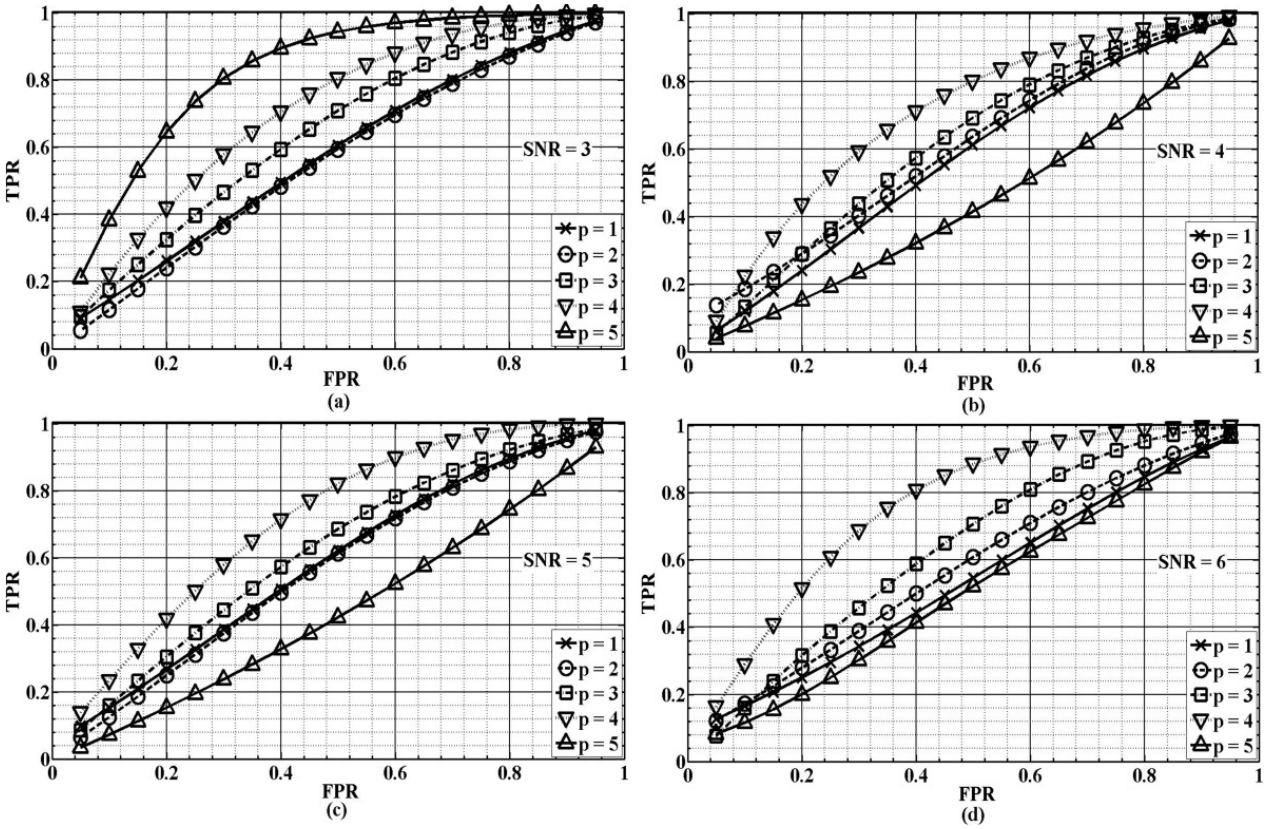


Figure 4.6 BBTM spike detection using STEO to generate spike templates with firing rate equaling 10, (a) SNR = 3, (b) SNR = 4, (c) SNR = 5, and (d) SNR = 6

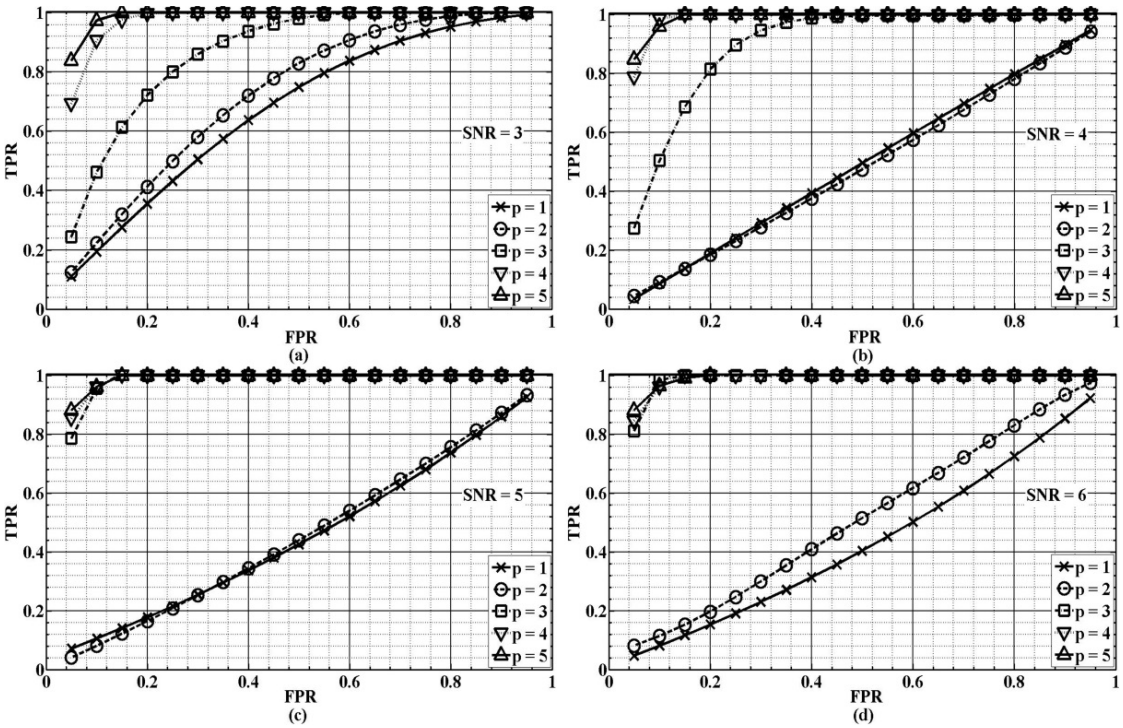


Figure 4.7 BBTM spike detection using STEO to generate spike templates with firing rate equaling 100, (a) SNR = 3, (b) SNR = 4, (c) SNR = 5, (d) SNR = 6

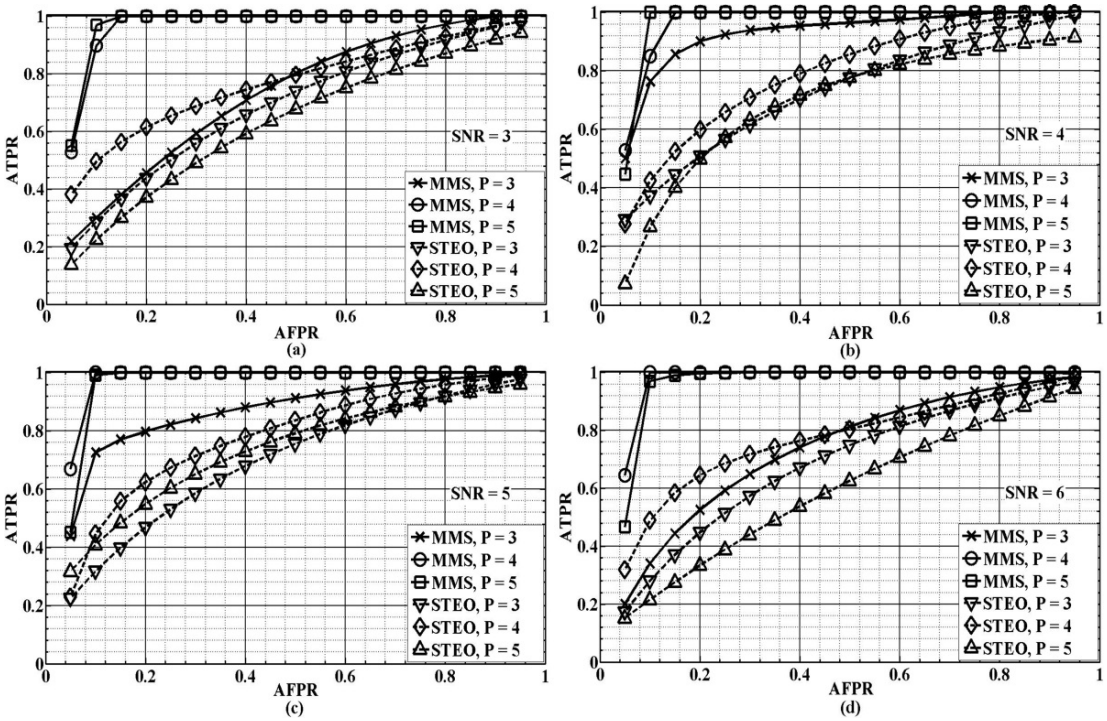


Figure 4.8 BBTM spike clustering with MMS-based and STEO-based spike generation methods, (a) SNR = 3, (b) SNR = 4, (c) SNR = 5, (d) SNR = 6

Moreover, the threshold control parameter is researched. From (4.14), TCP is a major parameter for detecting the spikes. In Figure 4.9, two parameters are used to research this parameter: FPR/TPR and TPR. It is ideal to find the lowest FPR/TPR with the highest TPR. Figures 4.9(a) and 4.9(b) are the results using known templates, and α can be chosen as 0.4 for the signals with the firing rates 10 and 100. Figures 4.9(c) - 4.9(f) show the results with unknown templates and using MMS methods to generate the templates. Figures 4.9(c), 4.9(e) and Figures 4.9(d), 4.9(f) are the signals with firing rates of 10 and 100, and the coefficient P equals to 4 and 5. From Figures 4.9(c) - 4.9(f), a P equaling 5 is more robust than the results of P equaling 4, because for all the signals with SNR 3-6, we can set a deterministic parameter, which is convenient for the circuit design. Comparing Figures 4.9(e) and 4.9(f), if we want to set TPR close to 1, then α can be chosen as 0.7, and for both firing rates and signals with an SNR from 3 to 6, the FPR/TPR are around 0.1 and 0.2.

4.4.4 Other Important Results and Discussions

The comparison between the original templates and generated templates is given. Figure 4.10 shows original and generated templates with SNRs 3-6 for three neurons. It can be found that the generated templates have shapes similar to the original templates.

The results of the spike detection and clustering for the signal with SNR 3 are given in Figure 4.11 and Figure 4.12. These two figures show the spike detection and classification results. It can be seen that for the signal with an SNR equaling 3, all the spikes are almost detected out, and that the BBTM method has good classification accuracy.

Finally, we compare our method with some other automatic template matching spike detection and clustering methods. Complexity and detection accuracy are compared among all these methods. Considering the complexity, it can be found that our proposed method has less calculation from Table 4.1, and the real computation time of the BBTM method is 1.2 ms. Also, comparing the TPRs and FPRs of the methods, when the SNR equals 3 and FPR equals around 0.03 respectively, BBTM method has a relatively higher TPR; therefore, BBTM method has comparatively low complexity and high detection accuracy.

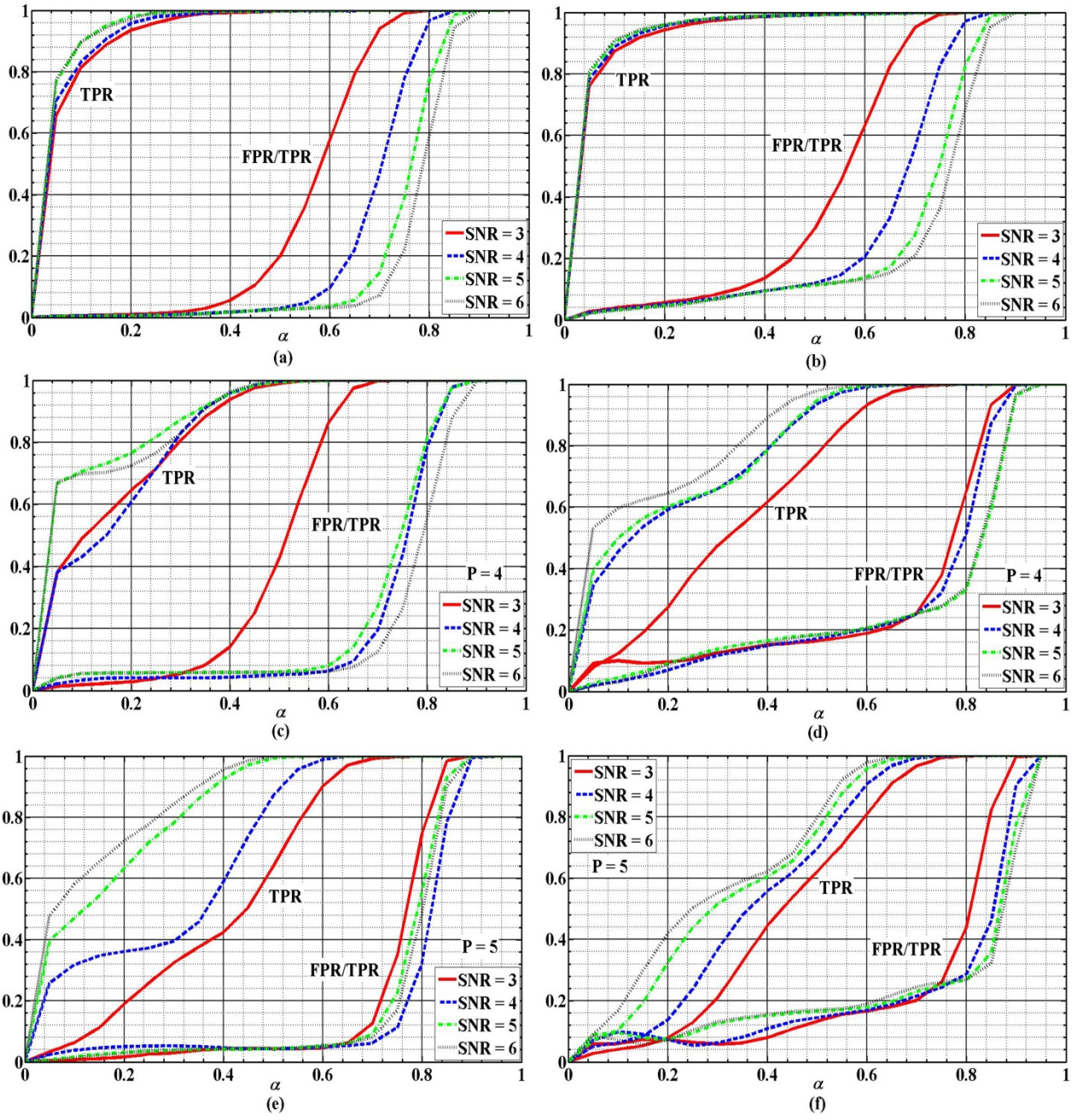


Figure 4.9 Research of the threshold control parameters, (a) firing rate is 10 with known templates, (b) firing rate is 100 with known templates, (c) firing rate is 10 with MMS template generation method when *P* equals 4, (d) firing rate is 100 with MMS template generation method when *P* equals 4, (e) firing rate is 10 with MMS template generation method when *P* equals 5, (f) firing rate is 100 with MMS template generation method when *P* equals 5

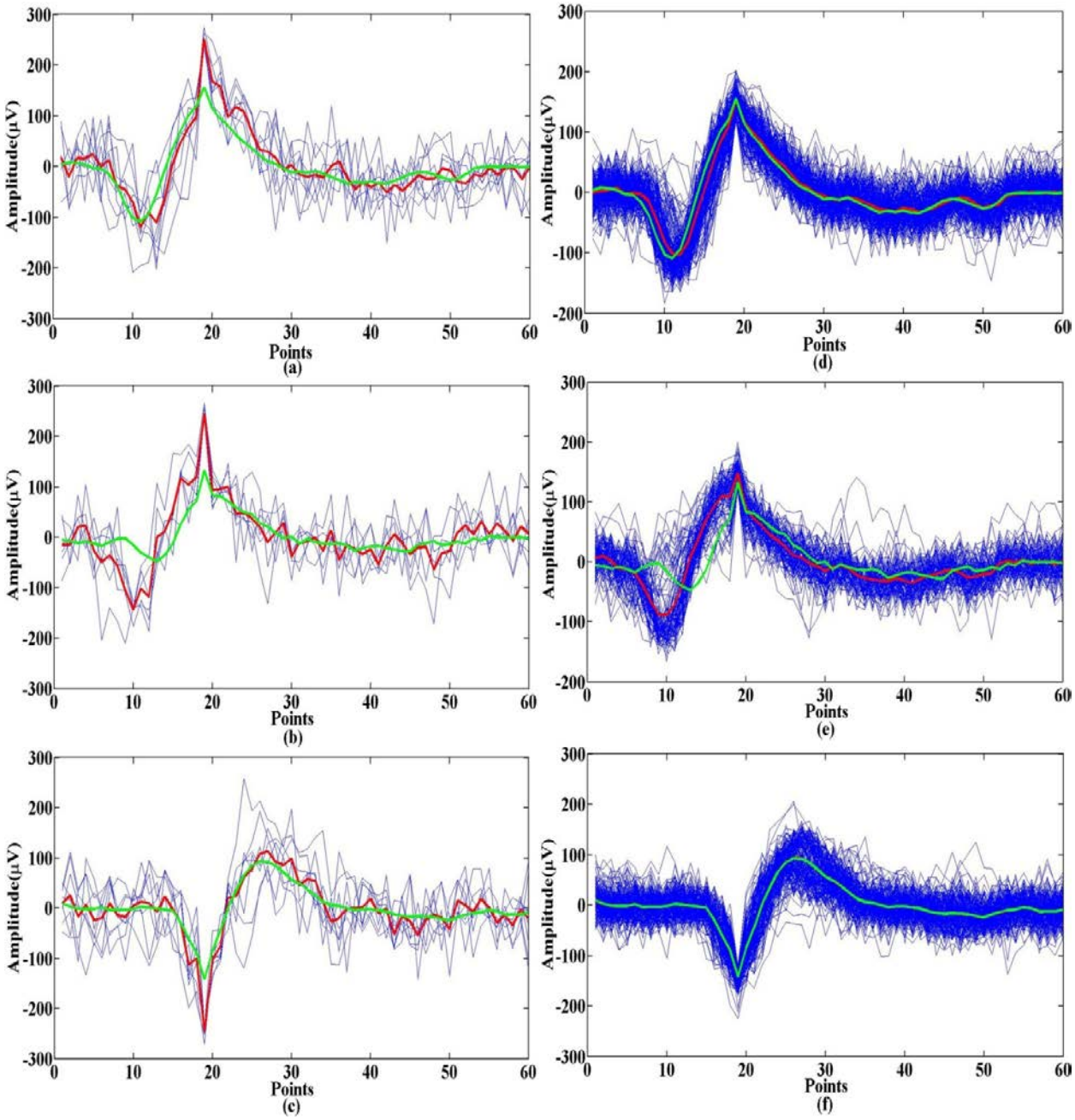


Figure 4.10 The results of the generation of the spike templates, (a)-(c) comparison between original and generated templates with signals SNR equaling 3, (d)-(f) comparison between original and generated templates with signals SNR equaling 6. The red color line is the generated spike templates and the green line is the original spike templates.

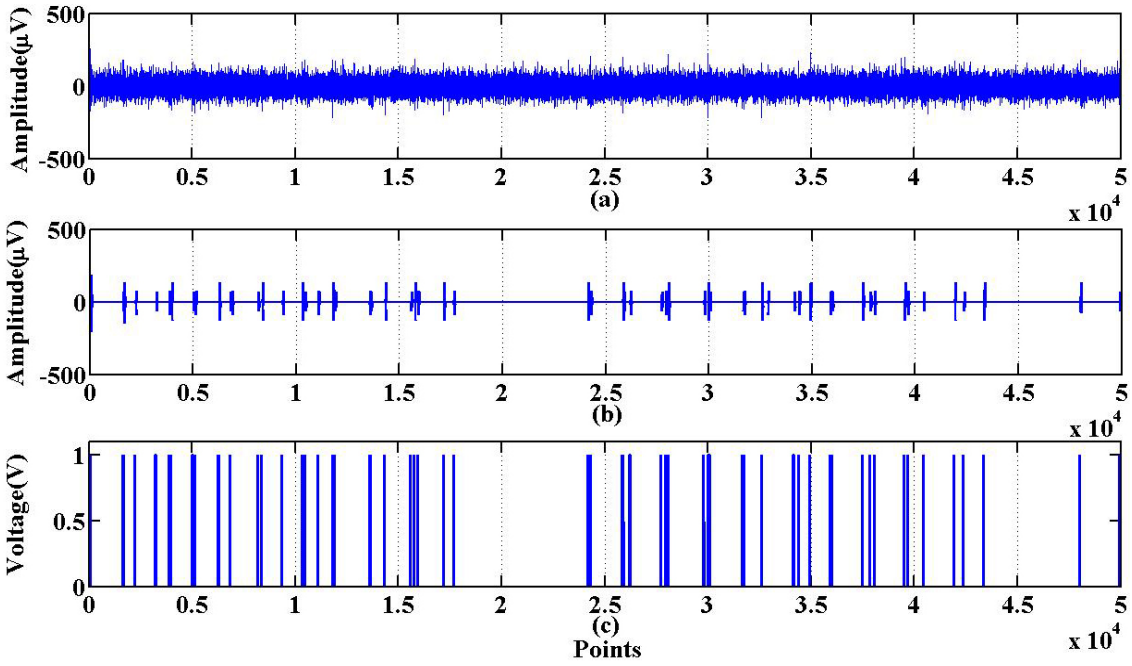


Figure 4.11 The detection results for the signal with SNR equaling 3, (a) original signals, (b) the spikes in the signal, (c) the detected spikes

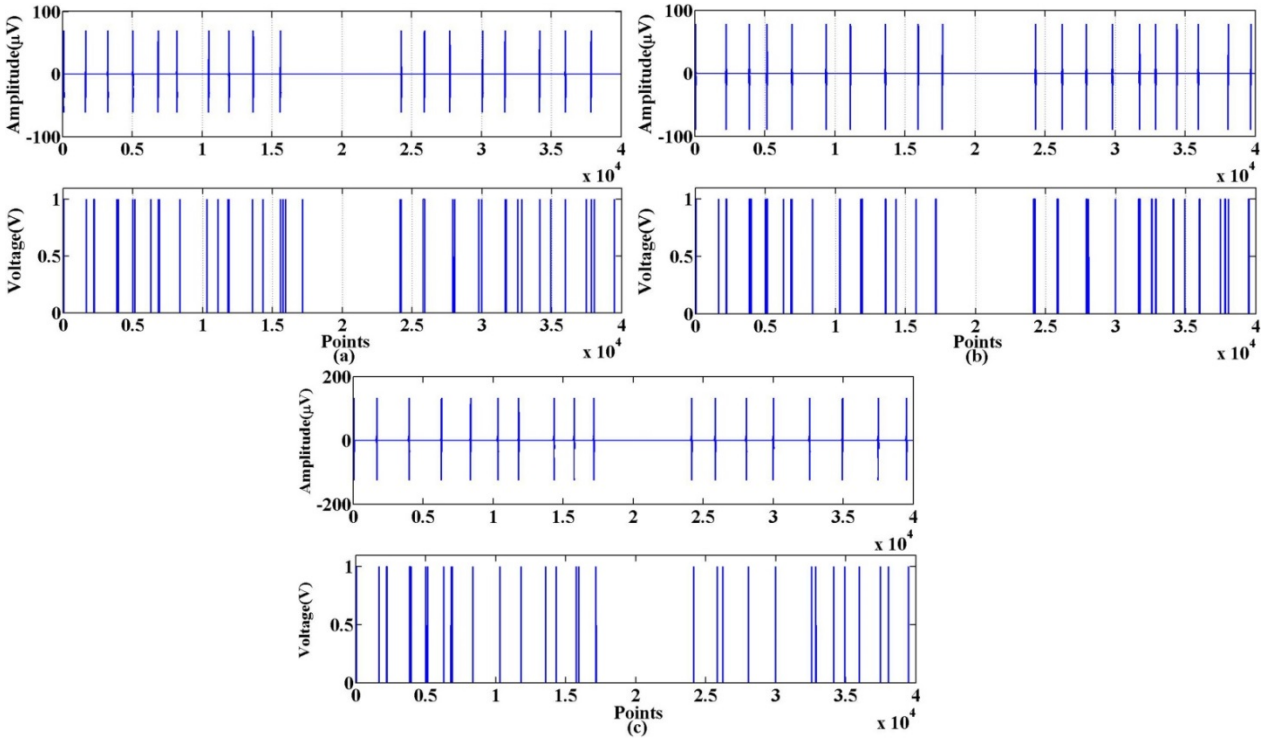


Figure 4.12 The comparison between the classified spikes and the original signals for the signal with the SNR equaling 3 for three neurons

Table 4.1 Comparison between the proposed BBTM method and other similar works

Ref.	Detection method	Complexity	Accuracy
[112]	Fast normalized correlator	(N+1)*multiplication + (N+1)*addition + 1*division + 1* squared root[a]	TPR : 0.84 (SNR =3) FPR : 0.01 (SNR =3)
[115]	M-sorter	Not given	TPR: ≈ 0.85 (SNR > 6) FPR : ≈ 0.2 (SNR > 6)
[211]	EC-PC spike detection	Not given	TPR: 0.2 (SNR = 3) FPR : 0.1 (SNR = 3) TPR: 0.95 (SNR = 6) FPR : 0.1 (SNR = 6)
[170]	Deconfusion method	P* (Number of neuron)^2 * length of the filter[a]	TPR : 0.86 (SNR =3) FPR : 0.04 (SNR =3)
This work	BBTM	N*multiplication + (N-1)*addition[a]	TPR : 0.90 (SNR =3) FPR : 0.04 (SNR =3) TPR : 0.90 (SNR =6) FPR : 0.03 (SNR =6)

[a] N is the number of the points, P is the coefficient

4.5 Conclusions

In this article, we described a Bayesian-based template matching spike detection system. This system not only can detect the spikes with known templates, but can also automatically generate spike templates to detect and cluster the spikes when the templates are unknown. We compared the proposed BBTM method with RMS, MAD, MMS, STEO, and S_STEO spike detection methods. It can be seen that the BBTM method has the best detection accuracy, and it can also automatically generate the spike templates using amplitude-based and energy-based methods. Based on the comparison, we chose the MMS method to generate the templates. Using the BBTM method, the TPR can reach up to 0.95 with an FPR of 0.05, which is a good detection accuracy. Additionally, the BBTM method provides spike classification. We used correlation and Euclidean distance to estimate the difference between the templates and the neural signals, and the thresholds are 0.8 and 0.5 respectively. Based on this method, the clustering accuracy can be around 1 when FPR equals 0.1 for three-neuron composite signals. The BBTM method has a short computational time, which is only around 1.2 ms for the spike detection and clustering of

each spike. Therefore, the BBTM method not only has a simple structure and low complexity, but also has accurate spike detection and classification.

CHAPTER 5 ARTICLE 3 : A DIGITAL MULTICHANNEL NEURAL SIGNAL PROCESSING SYSTEM USING COMPRESSED SENSING

Chapters 3 and 4 discuss the signal compression and reduction systems respectively, and it is necessary to further research the implementation of a neural signal processing system involving signal compression and reduction for an implantable neural recording interface.

In this chapter, we put forth a single and a multichannel system which includes signal compression and spike detection. The single-channel signal processing system is composed of spike detection and data compression blocks. The signal compression block applies the Minimum Euclidean or Manhattan Distance Cluster-based deterministic compressed sensing matrix that is proposed in chapter 3. The spike detection block uses amplitude-based spike detection, and threshold is calculated by root-mean-square method. For the construction of the MDC matrix, the distance σ is an important parameter, which can take a value of 4 or 5. In addition, based on the single-channel signal processing system, the sharing strategy is used to construct a multichannel system, and we analyze the influence of the number of the channels; scan rate on the reconstruction error, compression rate and power consumption; the influence of the signal-to-noise ratio; and reconstruction performance on neural signals. Based on the results, a 256-channel digital signal processing system, implemented in a 130-nm CMOS process, is proposed. This system has power consumption per channel of 12.5 μW and silicon area per channel of 0.03 mm^2, and provides data reduction of around 90% while enabling accurate reconstruction of the original signals.

(Digital Signal Processing, publication date: August 2016)

A Digital Multichannel Neural Signal Processing System Using Compressed Sensing

Nan Li[a*], Morgan Osborn[a], Guoxing Wang[b], and Mohamad Sawan[a]

[a] Polystim Neurotechnologies Lab. Electrical Engineering Dept., Polytechnique Montreal

2900 Edouard-Monpetit, H3T 1J4, Montreal (QC), Canada

[b] Department of Micro/Nano Electronics, School of Microelectronics, Shanghai Jiao Tong University Dongchuan Road #800, Minhang District, Shanghai, China 200240

ABSTRACT — This paper concerns a wireless multichannel neural recording system using a compressed sensing technique to compress the recorded data. We put forth a single and a multichannel system applying a Minimum Euclidean or Manhattan Distance Cluster-based (MDC) deterministic compressed sensing matrix. The single-channel signal processing system is composed of spike detection and data compression blocks. For the construction of the MDC matrix, the distance σ is an important parameter, which can take a value of 4 or 5. In addition, the sharing strategy is used to construct a multichannel system, and we analyze the influence of the number of the channels; scan rate on the reconstruction error, compression rate and power consumption; the influence of the signal-to-noise ratio; and reconstruction performance on neural signals. Based on the results, a 256-channel digital signal processing system, implemented in a 130-nm CMOS process, is proposed. This system has power consumption per channel of 12.5 μW and silicon area per channel of 0.03 mm^2, and provides data reduction of around 90% while enabling accurate reconstruction of the original signals.

Keywords — Multichannel neural recording, neural signal processing, data compression, compressed sensing, DSP, VLSI.

5.1 Introduction

Wireless monitoring of neural activity through implantable devices is an important technology that enables advanced diagnosis and treatment of brain disorders such as Parkinson's disease, major depressive disorder and epilepsy [212] [213] [214]. Figure 5.1 shows a typical wireless neural recording system. However, designing such a wireless neural recording device faces

numerous challenges. These include integrating high-density recording electrodes [215] [216], avoiding the heating of tissues due to energy transfer to power the implants (the maximum power density should be 0.8 mW/mm^2 for the exposed tissue area [178]), maximizing the device lifetime [57] [217], and minimizing the device size [218]. The conflict between huge data size and limited energy available for implantable recording devices is one of the principal challenges; specifically, integrating the necessary wireless transmission component in an implantable device exacerbates the problem of stringent energy constraints [134]. Therefore, data reduction or compression strategies should be employed to minimize the power consumption of the dedicated implantable devices.

Several neural signal reduction or compression techniques are already in use. Signal reduction is widely used to implement data reduction under certain constraints; methods include neural spike detection [90] [93] [95] and data feature extraction [88] [122]. Both methods involve locating important information and eliminating the remaining parts of the signals. However, signal reduction methods distort or lose some necessary information. For instance, a spike-detection-based neural recording device usually obtains data as the time series or the impulse, which cannot provide the details (shape or amplitude) of the original signal or spikes [148]; feature extraction methods are usually computationally complex, which conflicts with the design of a low-power device [105]. Therefore, it is necessary to find a new method that does not cause significant loss of features when recording neural signals.

Data compression methods avoid these drawbacks by preserving maximum information during the compression phase, which allows recovery of the original signal. Recently introduced compressed sensing (CS) technique shows great potential in compressing neural signals [131]. CS has low-encoder complexity and universality for different kinds of signals. It has attracted considerable attention in the areas of computer science, applied mathematics and electrical engineering [130]. CS preserves the temporal and morphological information of the signal, which is much better than spike detection or feature extraction methods [138].

5.1.1 Introduction of the CS Technique

In this section, we briefly introduce basic concepts in CS theory. First, the sparsity of the signal is an important concept. A sparse signal can be compressed through a sensing matrix. Suppose a vector (or signal) $x(x_1, x_2 \cdots, x_n) \in \mathbb{R}^N$ and some items of x are zero or close to zero, so this

vector can be called a sparse vector (or signal). If *x* is not sparse in the current basis, but it is sparse under some bases, then it still can be regarded as a sparse signal. For example, suppose a basis $\Psi_{N\times T}$, in which $x = \Psi z$ can be sparsely represented, so *x* is sparse under basis Ψ.

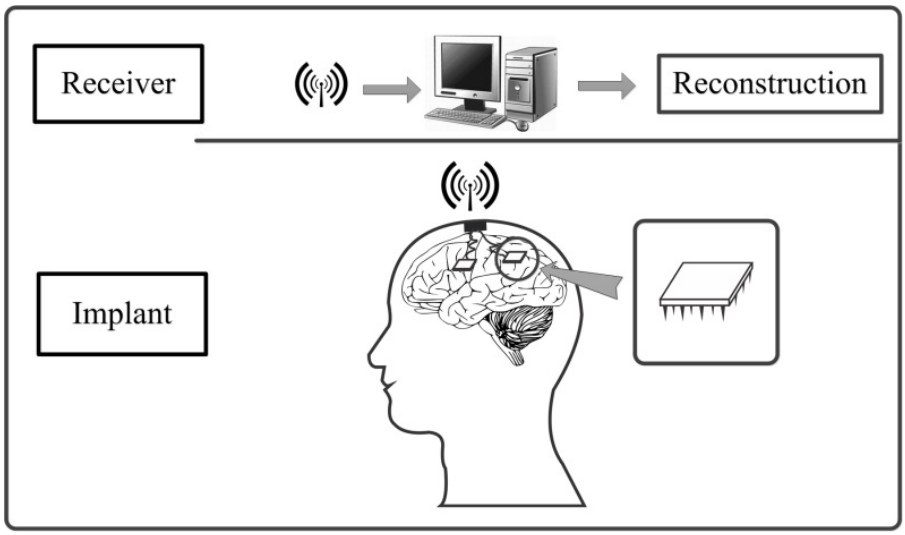


Figure 5.1 Simplified diagram of a typical wireless neural monitoring system

If $x(x_1, x_2 \cdots, x_n) \in \mathbb{R}^N$ is sparse, then it can be compressed through a sensing matrix $\Phi_{N\times M}$ to $y \in \mathbb{R}^M$. When the sparsity of the signal is large, *x* can be largely compressed, that is, $M \ll N$, which can be described as in (5.1).

$$y = \Phi_{N\times M} x \tag{5.1}$$

If *x* is sparse under basis Ψ, then (5.1) can change to be (5.2).

$$y = \Phi\Psi z \tag{5.2}$$

Second, the original signal can be reconstructed by ℓ_1 minimization. Given the original sparse signals and the measurement *y*, the best way to reconstruct the signal is through ℓ_0 minimization [130]. But finding a solution that approximates ℓ_0 minimization is NP (non-deterministic polynomial-time) hard; therefore, ℓ_1 minimization is widely used in signal reconstruction for CS application [130]. The form of ℓ_1 minimization is shown in (5.3). Based on the signal reconstruction via ℓ_1 minimization, many signal reconstruction algorithms exist. ℓ_1 minimization reconstruction algorithms, which directly use framework shown in (5.3), are powerful methods for computing sparse representations [136]; basis pursuit algorithms (BP) belong to this category [219]. Greedy algorithms are another category, which includes match pursuit algorithm (MP),

orthogonal matching pursuit algorithm (OMP) [181], and iterative hard or soft thresholding algorithm [182] [183]. Greedy algorithms are computationally efficient, but they are usually sensitive to noise especially when the original signals are not exactly sparse. By comparison, ℓ_1 minimization reconstruction algorithms are more robust to noise but at the price of a higher computational cost [220]. In addition, other kinds of algorithms can be used to reconstruct the original signals; for example, a Bayesian-based reconstruction method, called Block Sparse Bayesian Learning (BSBL) algorithm, uses the maximum likelihood to reconstruct the signal, and can reconstruct non-sparse signals [184].

$$x' = \underset{z}{\operatorname{argmin}} \|z\|_1 \quad \text{subject to} \quad z \in B(y) \tag{5.3}$$

where B(y) = $\{z : Az = y\}$.

Third, the design of the sensing matrix is another important topic. The sensing matrix strongly influences the amount of reconstruction error and also transmission of compressed signals [221]. In CS theory, the sensing matrix Φ can be a random matrix, such as a sub-Gaussian matrix [222], a random discrete Fourier transmission matrix [161], or a deterministic matrix, such as the Discrete Chirp matrix [163], the Reed Muller matrix [187], low-density parity-check (LDPC) matrix [155]. To correctly reconstruct x, the sensing matrix $\Phi_{N\times M}$ should obey the Restricted Isometry Property, which is described as follows.

Restricted Isometry Property An $M \times N$ sensing matrix Φ is said to satisfy Restricted Isometry Property (RIP) of *k order*, if it satisfies (5.4),

$$(1 - \varepsilon_k)\| X \|_2^2 \leq \| \Phi X \|_2^2 \leq (1 + \varepsilon_k)\| X \|_2^2 \tag{5.4}$$

for all the *k-sparse* vectors X. The restricted isometry constant ε_k of matrix Φ lies between 0 and 1. The process of CS compression is shown Figure 5.2. In this diagram, a sparse signal is firstly compressed by a sensing matrix. Then the signal is recovered through the ℓ_1 norm-based reconstruction. After the reconstruction, if x is sparse under the basis Ψ, it still needs to recover the signal in the current basis.

Finally, the research in the field of compressed sensing is not just in the theoretical concept but also in the design of underlying circuitry. There are several articles about the application of the CS technique [8] [185] [223]. Also, some designers used the CS technique to design the neural recording circuit [11] [134] [138]. Figure 5.3 shows the principles of use of the CS technique in

neural recording circuit design. Figures 5.3(a) and 5.3(b) depict analog and digital single-channel designs that apply the CS technique. The designs have two common parts: a sensing matrix generator and a multiplication block. In Figure 5.3, the sensing matrix generator could be a random or deterministic matrix (vector) generator, but most current designs use a random sensing matrix to design the circuit. The multiplication block does the matrix multiplication of the sensing matrix and the signal vector.

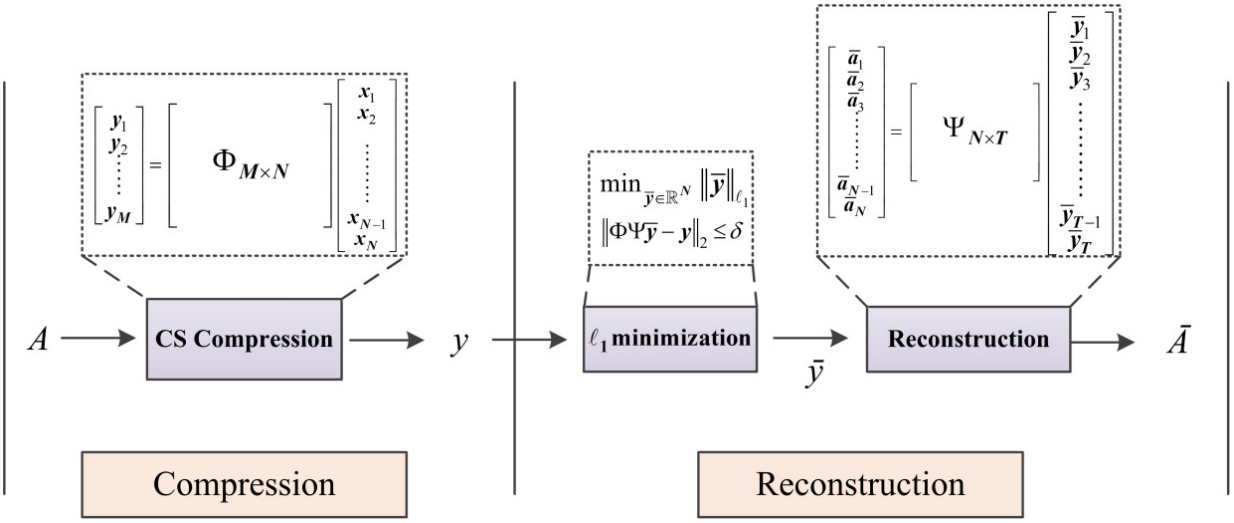


Figure 5.2 Framework of the compressed sensing technique

5.1.2 Contribution of This Article

In a recent article, we introduced a sensing matrix construction method called a minimum Euclidean or Manhattan distance cluster-based deterministic (MDC) sensing matrix [224]. We proved that the MDC matrix obeys the RIP under two prerequisites; also, we concluded that the MDC matrix can compress a signal with a relatively large compression rate (CR) and small reconstruction error rate (RER). We also previously proved that the MDC matrix can be used to compress signals whether the signal is sparse or not [224].

In this article, our contribution is using the MDC matrix to implement single and multi-channel digital systems. It can be seen from Figure 5.3 that the process of sensing matrix generation does not include any information from the signals that need to be compressed, but the MDC matrix can use the information of the signal. According to [224], we design a digital signal processing system which applies the MDC matrix; the principle of the circuit is shown in in Figure 5.3(c). The difference between our design and the ones in Figures 5.3(a) and 5.3(b) is that our design

uses the information of the signal itself to generate the sensing matrix. In later sections, we give details of construction of a digital circuit using the MDC matrix and discuss how to use the MDC matrix to design a multichannel signal processing system.

5.1.3 Structure of the Article

In the remaining parts of this paper, we briefly reiterate some concepts of the MDC sensing matrix for the construction of the sensing matrix in section 5.2. We introduce the simulation dataset in section 5.3. The circuit design and implementation are introduced in section 5.4. The simulation results and the discussion based on the design of the signal processing system are given in section 5.5. Finally, in section 5.6, we present our conclusions.

5.2 The Construction of the MDC Matrix

In this section, we briefly review concepts relating to the MDC matrix. The definitions of a *p*-dissimilar vector and the construction method of an MDC matrix, which are also discussed in [224], are given as follows.

Definition 1: (Equal Index Permutation) Suppose for a vector $X\left(x_1, x_2, \cdots, x_n\right)$, there exists a permutation $A_1\left(a_1, a_2, \cdots, a_t\right)$ of the index vector $(1, 2, \cdots, n)$, and a vector based on this index permutation $X_{A_1}(x_{a_1}, x_{a_2}, \cdots, x_{a_t})$. If every two items from X_{A_1} are identical under some measures, specifically, $x_{a_i} = x_{a_j}$, $x_{a_i}, x_{a_j} \in X_{A_1}$, A_1 is called an equal index permutation. If this measure is a Euclidean (or Manhattan) measure, A_1 is called an equal index permutation under a Euclidean (or Manhattan) measure.

Definition 2: Suppose for a vector $X\left(x_1, x_2, \cdots, x_n\right)$, there exists an index set containing M equal index permutations under a Euclidean (or Manhattan) measure, i.e. $A_M\left(A_1, A_2, \cdots, A_m\right)$, where $A_i = (a_{i_1}, a_{i_2}, \cdots, a_{i_t})$, $a_{1_t} \in (1, 2, \cdots, n)$. X can be clustered into M clusters according to the index A_M, that is, $X_{A_M}(x_{A_1}, x_{A_2}, \cdots, x_{A_m})$. If

1. If $\forall x_i \in X, x_i \in x_{A_i}$ and $x_i \notin x_{A_j}, A_i, A_j \in A_M$, then $A_i \neq A_j$.
2. If $\forall x_i, x_j \in X, x_i \in x_{A_i}, x_j \in x_{A_j}$ and $x_i \neq x_j, A_i, A_j \in A_M$, then $A_i \neq A_j$.

So A_M is called an exclusive equal index permutation set, and vector X is called an M-cluster exclusive vector under permutation set A_M. This definition ensures that each point is clustered into a unique cluster, and also two identical points need to be clustered into the same cluster.

Definition 3: (p-dissimilar vector) Suppose a vector $X\left(x_1, x_2, \cdots, x_n\right)$ is an M-cluster exclusive vector under permutation set $A_M(A_1, A_2, \cdots, A_m)$, where $A_i = (a_{i_1}, a_{i_2}, \cdots, a_{i_t})$, $a_{1_t} \in (1, 2, \cdots, n)$. According to definition 2, X can be clustered into M clusters based on the index A_M, that is, $X_{A_M}(x_{A_1}, x_{A_2}, \cdots, x_{A_m})$. Letting $p = M$, vector X can be called a p-dissimilar vector. The size of each cluster $C_{x_{A_i}}$ is $\mathrm{I}(C_{x_{A_1}}) = t$, $X_{A_1} \in X_{A_M}$. So $C_{x_{A_i}}$ is called a t-large cluster. If $t = 1$, $C_{x_{A_i}}$ is called a unit-large cluster. If A_i, $\forall A_i \in A_M$, is an equal index permutation under a Euclidean (or Manhattan) measure, X is called a Euclidean (or Manhattan) measure p-dissimilar vector.

Definition 4: (*Equivalent Index Subset Vector*) Suppose two vectors $X\left(x_1, x_2, \cdots, x_n\right)$ and $Y\left(y_1, y_2, \cdots, y_n\right)$ have the same length $\mathrm{L}(X) = \mathrm{L}(Y) = n$. X is an M-cluster exclusive vector under permutation set $A_M(A_1, A_2, \cdots, A_m)$, where $A_i = (a_{i_1}, a_{i_2}, \cdots, a_{i_t})$, $a_{1_t} \in (1, 2, \cdots, n)$. For a determined subset A_i, $A_i \in A_M$, if $\{y_{a_i} = r \mid a_i \in A_i\}$ and $\{y_{a_i} = 0 \mid a_i \notin A_i\}$, Y is called an equivalent index subset vector of the vector X.

When $r = 1$, Y is called the unit equivalent index subset vector; when $r = r/\ell_2(r)$, it is called the normalized equivalent index subset vector. For a p-dissimilar vector X, there are M equivalent index subset vectors.

Definition 5: *(Minimum Euclidean or Manhattan distance cluster-based deterministic sensing matrix (MDC matrix))*

If a vector X is a Euclidean (or Manhattan) measure p-dissimilar vector, we can construct a deterministic sensing matrix through the three following steps.

(Step1).Divide X into M dissimilar clusters $\{C(x_{A_1}), C(x_{A_2}), \cdots, C(x_{A_m})\}$ based on the exclusive equal index permutation set $X_{A_M}(x_{A_1}, x_{A_2}, \cdots, x_{A_m})$.

(Step2).The equivalent subset index vector of these clusters $\{C(x_{A_1}), C(x_{A_2}), \cdots, C(x_{A_m})\}$ is $\{\phi_1, \phi_2, \cdots \phi_m\}$, $\mathrm{m} \in \mathbb{N}$.

(Step3).Compose the matrix with $\{\phi_1, \phi_2, \cdots \phi_m\}$, $\mathrm{m} \in \mathbb{N}$, which is $\Phi = [\phi_1\,;\ \phi_2\,;\ \cdots\ ;\ \phi_m]$.

Thus, Φ is called a minimum Euclidean or Manhattan distance cluster-based deterministic sensing (MDC) matrix. If all of the ϕ_i , $i \in [1, m]$ in Φ are the normalized equivalent index subset vectors, Φ is called a normalized MDC (NMDC) matrix . If all ϕ_i , $i \in [1, m]$ in Φ are the unit equivalent index subset vectors, Φ is called a unit MDC (UMDC) matrix.

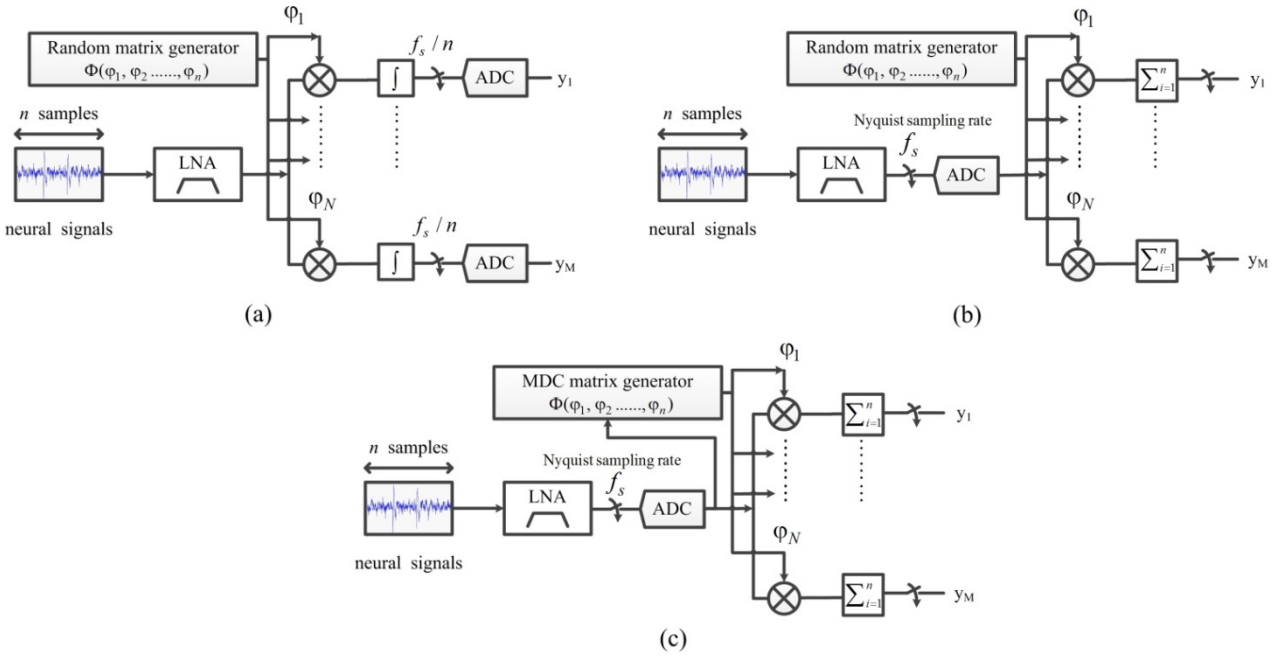


Figure 5.3 Diagram of the circuit design using the CS technique: (a) analog design, (b) digital design, (c) proposed digital circuit design using the MDC matrix

Given a vector X (x_1, x_1, x_2, x_3 , x_1 , x_2), (1,2,5) and (3, 6) are two equal index permutations. According to definition 2, this vector can be clustered into three clusters $\{x_1, x_1, x_1\}, \{x_2, x_2\}, \{x_3\}$ and its correspondent A_M is $\{(1,2,5), (3,6), (4)\}$. X is called a 3-dissimilar vector. The normalized equivalent index subset vectors of the clusters $\{x_1, x_1, x_1\}$, $\{x_2, x_2\}$, $\{x_3\}$ are $\{(1/\sqrt{3}, 1/\sqrt{3}, 0, 0, 1/\sqrt{3}, 0), (0, 0, 1/\sqrt{2}, 0, 0, 1/\sqrt{2}), (0, 0, 0, 1, 0, 0)\}$, and the NMDC matrix, Φ, for the vector X is shown in (5.5).

$$\Phi = \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 0 & 0 & 1/\sqrt{3} & 0 \\ 0 & 0 & 1/\sqrt{2} & 0 & 0 & 1/\sqrt{2} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.5}$$

5.3 Materials and Methods

All the algorithms, methods and data analysis procedures were implemented in MATLAB (Mathworks, Natick, MA). The circuit was described in Modelsim (Mentor Graphics, Wilsonville, OR) and the post-layout of the circuit was designed in Cadence Encounter (Cadence Design Systems, San Jose, CA). The power consumption and silicon area of the circuit were estimated by Synopsys (Synopsys, Mountain View, CA). We used three datasets to make the simulation.

The first dataset was acquired from the prefrontal cortex of an adult male rhesus macaque monkey (Cognitive Neurophysiology Laboratory, McGill University). The recording circuit contained 32 extracellular channels with a Utath 10 × 10 microelectrode array. The data were comprised of three different recordings over three trials. The duration of each trial was 300 s. The data were filtered initially with a third-order bandpass Butterworth analog filter with cutoff frequencies of 0.3 and 7 kHz. Then, the filtered data were amplified with an 80 db gain amplifier, sampled at 30 kHz and digitized at 10 bits per sample.

The second dataset was obtained from the visual cortex of a rat (Center for Studies in Behavioral Neurobiology, Concordia University). The researchers used a stainless-steel-tipped microelectrode with a shank diameter of 75 μm to record the data. The duration of the recording was 60 s. The data were filtered with a fourth-order bandpass Butterworth analog filter and the cutoff frequencies were between 150 Hz and 10 kHz. After filtering, the data were amplified with a gain of 100 db, sampled at 32 kHz and digitized at 10 bits per sample.

The third dataset came from a human medial temporal lobe (NeuroEngineering Lab, University of Leicester [190]). The dataset was acquired by using intracranial electrodes. The duration of the signal was ten seconds long. The data were sampled at 32 kHz, filtered between 300 Hz and 3 kHz and digitized at 12 bits per sample.

To imitate similar recording conditions, the datasets were refiltered with a fourth-order non-causal Butterworth high-pass digital filter with a cutoff frequency of 300 Hz and resampled at 25 kHz. Then, we randomly selected five groups of test data from the three datasets and ensured that data from every set were used. The circuit was designed with VHDL in Modelsim. Then we employed two different methods to acquire the results: the first involved designing the post-synthesis circuit (Synopsys) and post-layout circuit (Cadence Encounter), the other involved a Xilinx Virtex-6 FPGA ML605 evaluation board. The area of our designed digital circuit was

estimated by Synopsys Design Vision. The power consumption was estimated by Synopsys Design Vision and Prime Time. The library for simulation was IBM CMOS130 nm at room temperature 25℃ and the voltage was 1.2 V.

We used two methods to measure power consumption accurately. We first used Design Vision to estimate the power consumption, then generated 16 groups of the test benches to imitate different input data under different scan rates. All the input data were used to generate value change dump (VCD) files which contain the signal activities in one second. Then the VCD files and the circuit description were input to Prime Time to generate the power consumption. Finally, we compared and synthesized the power consumption estimates generated by two methods to produce a reliable estimate of power consumption under different scan rates, and analyzed the relationship between the scan rate and the power consumption.

The reconstruction algorithms used in this article were BP and Least Absolute Shrinkage and Selection Operator (Lasso) algorithms. BP , using default values, is from [192]. Lasso, using default values, is from [193].

The definitions of RER and CR are given in (5.6) and (5.7),

$$\mathrm{RER} = \|\nabla(\Phi x) - x\|_2 / \|x\|_2 \tag{5.6}$$

$$\mathrm{CR} = 1 - (N/M) \tag{5.7}$$

where $\Phi_{M \times N}$ is the sensing matrix with M rows and N columns, $\nabla(\Phi x)$ is the reconstruction of the original signal and x is the signal.

To study the influence of the signal-to-noise ratio (SNR), we built signals with different SNRs. We first extracted the neural spikes. The spikes were selected from 32 groups of real neural signals from our first dataset. We detected spikes, then performed feature extraction and clustering. At last, we selected five different groups (templates) of spikes to build composite signals. The length of a neural spike series is 2 ms (48 samples per spike). After the construction of the spike series, we inserted them into Gaussian background noise. Signals were achieved by inserting one unique or composite spike template(s) into the background noise with a Poisson firing model using a refractory period of 2 ms, and the firing rate was set at 100 spikes per second. Finally, we built four groups of signals with an SNR (see (5.8)) from 3 to 6; every group contained 32 signals to imitate signals from 32 different channels. This definition of SNR is a

good choice for the estimation of spike detection, because it prevents errors from the relative frequency of spikes of different amplitudes [90] [96].

$$\text{SNR} = \frac{\text{Maximum magnitude of the spike waveform}}{\text{Standard deviation of background noises}} \tag{5.8}$$

5.4 Circuit Design and Implementation

In this section, we describe the design of a CS-based digital signal processing circuit. A top-level view of the design is initially introduced, then the design of a spike detection block is reported, followed by the design of the data compression block using the MDC matrix, and the design of a multichannel system is detailed.

5.4.1 Single-channel Digital Data Compression System

The diagram of a neural recording circuit is shown in Figure 5.4. The system can be divided into three parts: front-end (mainly amplifier and filter), signal processing module and transmitter block. Our design is focused on the signal processing module, which is composed of the spike detection and data compression blocks.

5.4.2 Spike Detection Block

The spike detection block, shown in Figure 5.5, is used in deterministic single-channel processing. Both detection and compression blocks work in the single-channel processing, but for the multichannel processing, only the data compression block works.

For the spike detection block, we chose the root mean square (RMS) method to calculate the standard deviation (SD). We compared the RMS method with the median absolute deviation (MAD) method [98], maximum and minimum spread (MMS) sorting method [96], and smooth Teager energy operator (STEO) method [90] in terms of power consumption, estimation accuracy and complexity. We found that RMS has the lowest power consumption, the lowest complexity and an acceptable detection performance. Additionally, the system includes the data compression block, and spikes can also be detected through the reconstructed signal. Therefore, we chose the RMS estimator as the detection block. The spike detection block contains three major parts:

1) The standard deviation calculation. The calculation is based on (5.9).

$$SD = \sqrt{\frac{1}{N}\sum_{1}^{N}(x_n - \tilde{x})} \tag{5.9}$$

where x_n is the data point and $\tilde{x}$ is the expected value of the data. The circuit design contains an adder, a multiplier, a square root calculator and a shifter.

2) After calculating SD, the threshold is acquired. The calculation of the threshold is based on (5.10).

$$T = P \times SD \tag{5.10}$$

where P is the threshold coefficient, which can be 2 ~ 6. Based on the experimental comparison, we chose 3 for the design (the user can choose P for their specific usage).

3) When the threshold is computed, the detection is carried out by the spike detector. In our system, we use a two-bit detection code to express the results shown in Table 5.1. With the detection code, the spike can be found through a spike-analysis algorithm in a computer.

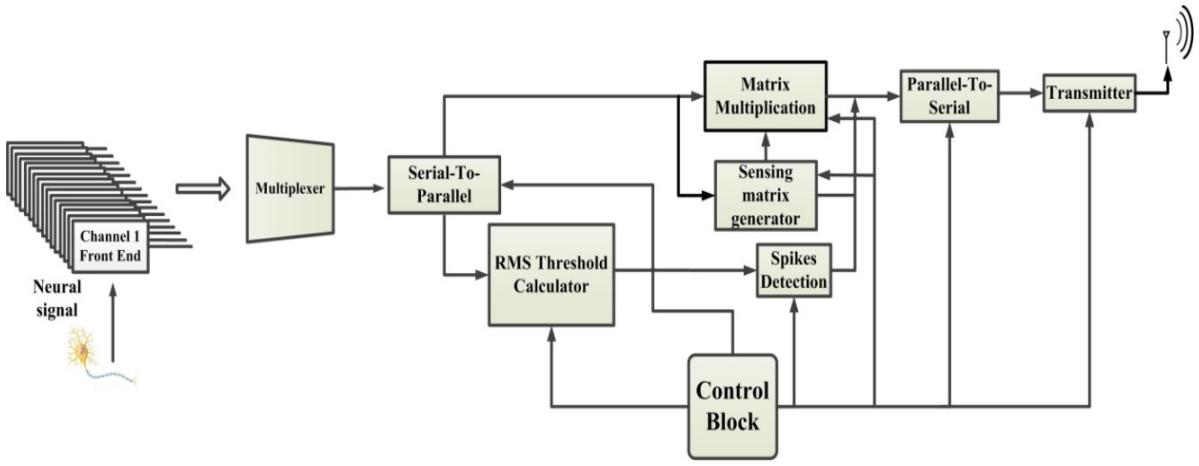


Figure 5.4 Diagram of the design of digital single-channel circuit

5.4.3 Data Compression Block

The data compression block can be divided into two major parts: the MDC matrix generator and the matrix multiplication block shown in Figure 5.6.

The MDC matrix generator is an important part of the proposed system. In [224], we described an algorithm, called the core data clustering algorithm, used to construct the MDC matrix. The

core data clustering algorithm is shown in Figure 5.6(a) and the data compression block is based on this algorithm. This algorithm has two important parameters: the core data (in Figure 5.6(a), the core data is the first data added into a new cluster) and the distance σ. Figure 5.6(b) and 5.6(c) show the behavior diagram and the implemented structure of this algorithm. The mathematical expression of σ is in Figure 5.6(a), which is $\|x_i - x_j\| \leq \sigma, x_i, x_j \in x(x_1, x_2 \cdots x_n)$. σ determines the number of the clusters and the sensing matrix for a specific signal. The whole block is composed of the comparator, the cluster indicator and the decoder. When new data are input, the enabled comparators make the comparison between the input x and the core data. If x is the first piece of data, then it is regarded as core data and put into the first cluster; if the input x is inside the range of the core data under the indicated distance, then it is clustered into the corresponding clusters. Supposing a unit of core data c_i of the cluster C_i and distance l, if $x \in (c_i - \mathrm{l}, c_i + \mathrm{l})$, x is clustered into the cluster C_i and the output of the cluster indicator is 1, or else, the output is 0. If x can be clustered into several clusters, for example, (C_i , C_{i+1} , C_{i+2} , ……, C_j), then x is clustered into the first cluster C_i . If x cannot be clustered into the existing clusters, then a new cluster is created with x as the core data.

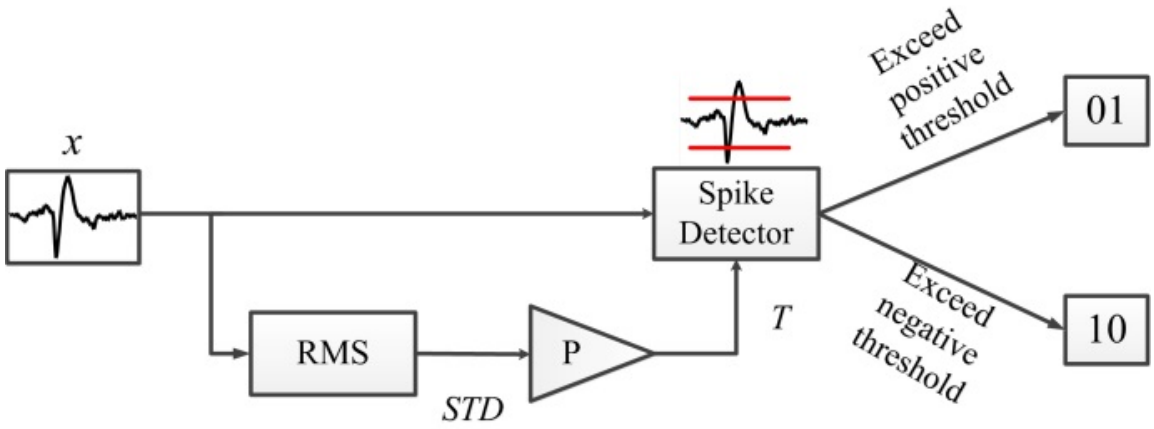


Figure 5.5 Diagram of the spike detection block

Table 5.1 Detection code of the spike detector

Code	**00**	**01**	**10**	**11**
Meaning	normal points	> positive threshold	< negative threshold	no detection

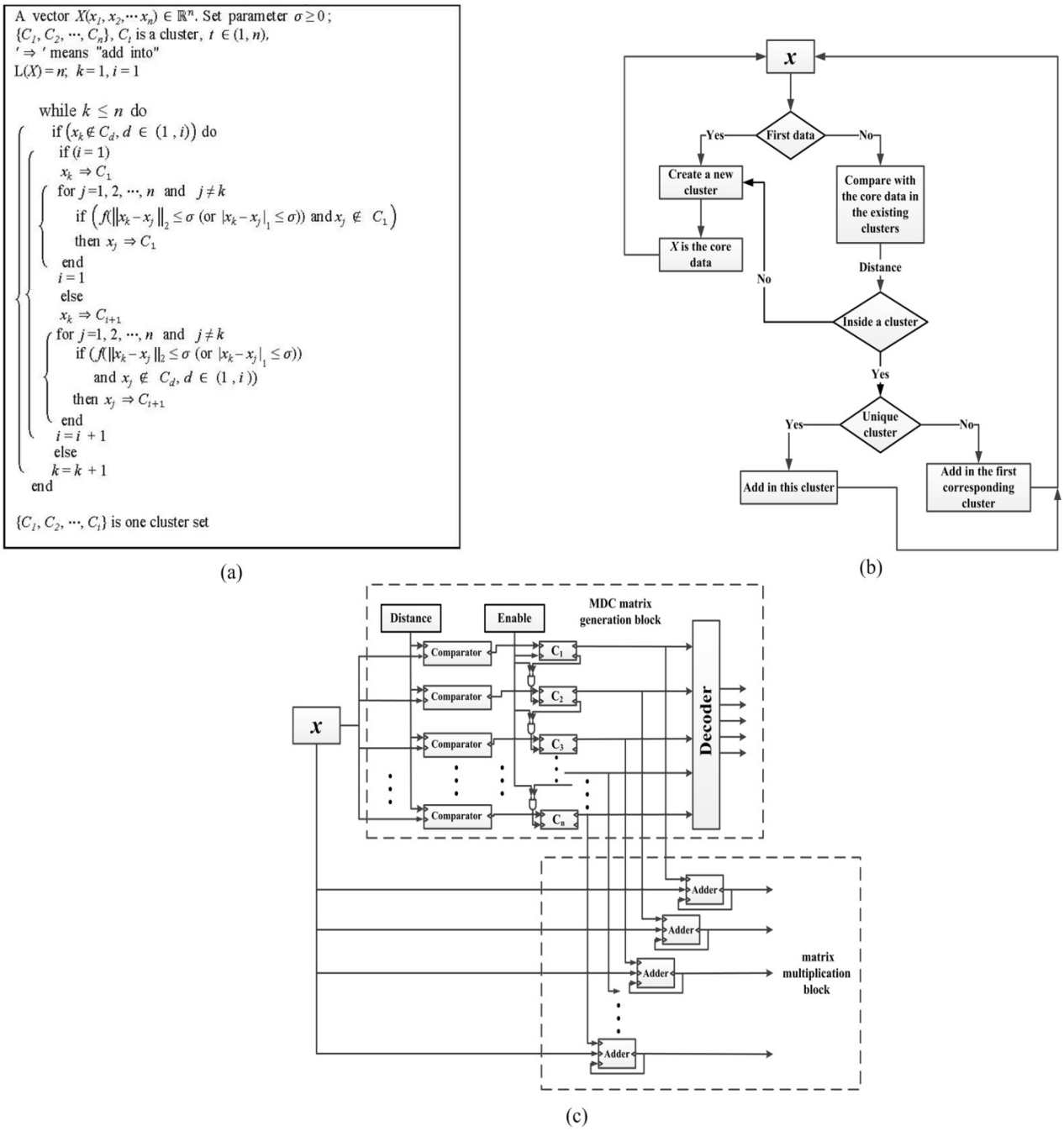


Figure 5.6 Design of the data compression block: (a) core data clustering algorithm [224], (b) behavior diagram of the core data clustering algorithm, (c) diagram of the digital circuit of the core data clustering algorithm

After the construction of the MDC matrix, a multiplication block is needed to compress the data. This block, shown in Figure 5.6(c), implements the multiplication between a signal vector and the MDC matrix. After compression by an $N \times M$ sensing matrix, an N-length vector becomes an

M-length vector, which completes the compression. We proved in [224] that the UMDC matrix can be efficiently used to compress signals. Because the UMDC matrix consists of zeros and ones, it can greatly reduce the complexity of the system (i.e. the power consumption and the area of the designed circuit). The multiplication block is mostly composed of adders to fulfill the matrix multiplication.

5.4.4 Multichannel Signal Processing

Based on the discussion of the single-channel processing method, in this sub-section, we introduce our method of construction of a multichannel system. The multichannel system has two modes: the first is for signal processing inside a single block (scan mode I), and the second is for processing between different blocks (scan mode II). In Figure 5.7, we give an example of a 256-channel system. These 256 channels are divided into 16 main blocks and each block comprises 16 channels. Scan mode I mainly records signals from a small volume of neurons, that is, one block. In this mode, a block is first chosen from block A to P. Then, a scan rate is selected to read the input data in this selected block. The second mode is scan mode II. This mode is used to record signals from different blocks, and it is mainly for the recording of signals from a relatively larger area or longer distance. In this mode, it needs to select one channel from each 16 blocks (16 channels from block A to P), for example, choosing A_{11}, B_{21}, …, P_{23}. Then the user chooses one scan rate to simultaneously record signals from these channels. Finally, the channels inside a block and the number of the blocks must be chosen. The number of channels inside a block should be two to the n^{th} power and each block has the same number of channels. The number of blocks should be two to the m^{th} power. The number of the channels can be calculated by (5.11).

$$\text{Total channels} = 2^n \times 2^m, \quad n, m = 1, 2 \cdots \cdots T, T \in \mathbb{N} \tag{5.11}$$

where *n* is the inside-block channel control parameter and *m* is the block control parameter; 2^n is the number of the channels in one block and 2^m is the number of the blocks.

Because we use the sharing strategy to construct a 256-channel system, it is important that both modes use the scan. The scan refers to the quantity of data used at the input of the system during a period of time. Scan has two aspects: scan direction and scan rate. The scan direction is periodically from left to right, and also from top to bottom (e.g. in Figure 5.7, A11 → A14 → A24 → A21 → A31→ A34 → A44 → A41 → A11 for scan mode I and A → D → H → E → I

→ L → P → M → A for scan mode II). This scan direction is used for both scan modes. Scan rate is related to the sampling rate of one channel and can be chosen by (5.12).

$$\text{SR} = sampling\,rate \times 2^r,\ r \le n\ or\ m \tag{5.12}$$

where *r* is the scan rate control parameter, *n* is the inside-block channel control parameter and *m* is the block control parameter. In scan mode I, *r* should be small or equal to *n*. In scan mode II, *r* needs to be smaller than *m*. Under both scan modes, the designer can choose different *r* to adjust the scan rate. Following from the scan modes, the channel-to-scan (ChS) parameter, shown in (5.13) needs to be introduced.

$$\text{ChS} = \frac{2^t \times sampling\,rate}{\text{SR}},\quad t = n, m \tag{5.13}$$

where SR is the scan rate, *n* is the inside-block channel control parameter and *m* is the block control parameter. In (5.13), *n* and *m* are for scan mode I and II respectively. These parameters can be used in the following sections to estimate the sampling rate and the number of channels or blocks.

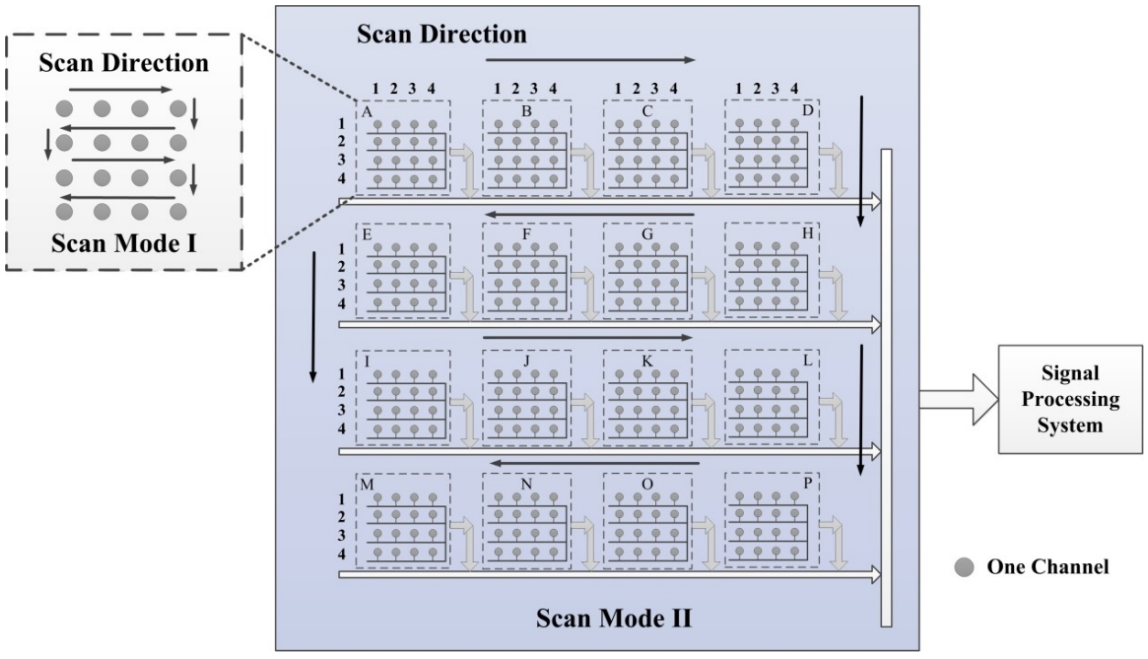


Figure 5.7 Diagram of the multichannel system

5.5 Results and Discussion

In this section, main parameters related to both single-channel and multichannel designs are highlighted. Then, the results of multichannel simulation and the specific post-layout circuit are discussed. Finally, the achieved design is compared to similar work in the literature.

5.5.1 Single-channel Data Compression System

In the core data clustering algorithm, the distance between the current data point and the core data point is an important parameter. This parameter determines the compression rate and reconstruction error. In Figure 5.8, the relationship between the distance and the compression rate, and the relationship between the distance and the reconstruction error are presented. Figure 5.8 (a) shows the relation between the reconstruction error and the distance under two introduced reconstruction algorithms above: BP and Lasso algorithms. In [224], we showed that the BP and Lasso algorithms are the two best algorithms to use when constructing the original signal compressed by the MDC matrix, and now we compare both algorithms with varying distances. Figure 5.8 (a) demonstrates that with increasing distance, the reconstruction errors under the two reconstruction algorithms show a nearly linear increase.

Lasso and BP algorithms have similar reconstruction performance at different distances. The reconstruction error under the BP algorithm is slightly smaller than that under the Lasso algorithm when distance is greater than 10, but the difference is not obvious when distance is less than 10. Effectively, both methods can be used to reconstruct original signals. In addition, Figure 5.8(a) shows that when distance equals 4 or 5, the RER is around 0.2, so if a minimal reconstruction error is necessary, then the distance should be smaller than 5.

The relationship between the compression rate and the distance requires explanation. Figure 5.8 (b) shows two cases: the compressed data alone and the compressed data with the sensing matrix. If we only consider the compressed data, the compression rate of the signal can be up to 99%, but when using CS, the unit usually needs the sensing matrix to reconstruct the original signals, that is, the same MDC is needed to recover the signal, so the sensing matrix has to be transmitted. In our implementation, we need to transmit the sensing matrix out. If we take into account the transmission of the sensing matrix, the compression rate can be up to around 60%. Considering data with a resolution of 10 bits and a distance exceeding 6, in Figure 5.8 (b), the compression

rates are close to 90% and 60% when considering only compressed data and compressed data with the sensing matrix, respectively. After evaluating the compression rate and reconstruction error, we can conclude that the distance should be chosen as 4 or 5. Under either value, the compression rate can exceed 50% and the reconstruction error is only around 0.2 using 10-bit data points. In our design, we chose 4 as the distance for construction of the MDC matrix.

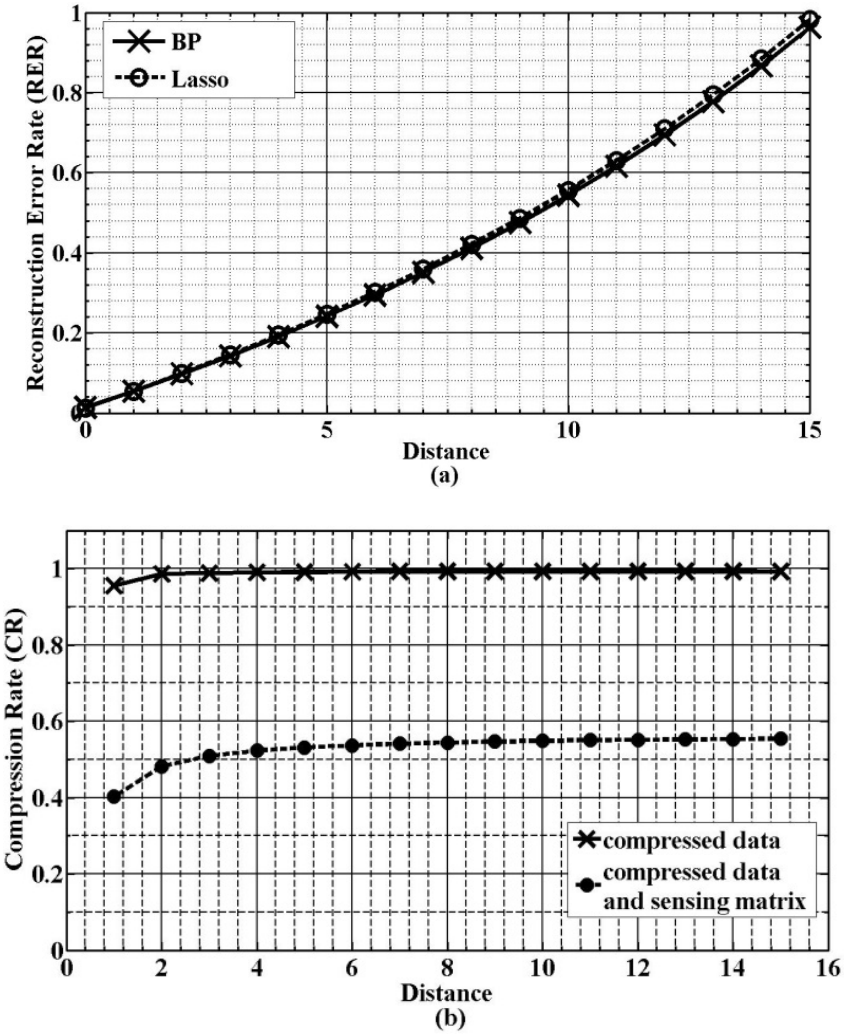


Figure 5.8 Relation between the distance and reconstruction error, compression rate: (a) relation between the distance and the reconstruction error using BP and Lasso algorithms, (b) relation between the distance and the compression rate

5.5.2 Multichannel Signal Compression System

It is important to note that the compression block can be shared by several channels, so some parameters for the design of the multichannel system need to be discussed. As explained above,

the multichannel system is mainly controlled by SR and ChS. We are mainly concerned with the relation between these two parameters and CR, RER and power consumption. In fact, the reconstruction error does not fluctuate much with the compression rate under different ChS. Moreover, when ChS increases, the reconstruction rate also increases. In Figure 5.9, when ChS equals 1, which means that the scan rate is equal to the maximum frequency ($2^n \times \text{sampling rate}$ or $2^m \times \text{sampling rate}$) and all the data can be recorded and input into the signal processing system, the RER remains at the minimum value (less than 0.1). When ChS equals 2 and the scan rate is half of the maximum scan rate, the RER reaches around 35%. When the scan rate reaches one-fourth and one-eighth of the maximum scan rate, the RERs are around 70% and 90% respectively. When increasing the scan rate to 1/16 and 1/32 of the maximum SR, the RERs exceed 1, which is not appropriate for use. Therefore, a ChS equals to 1,2,4 or 8 can be considered for designing the multichannel system. We used BP and Lasso algorithms to recover the original signals. We found that the algorithms give similar reconstruction performance: in Figure 5.9(a)-(b), the RERs are very similar under the same CRs. Therefore, reconstruction error increases with the increase of ChS, and BP and Lasso algorithms can both be applied to recover the multichannel signals.

Moreover, the influence of the SNR is researched. The relationship between the ChS, SNR and RER is shown in Figure 5.10. Figure 5.9 shows the RERs are steady when the compression rates are between 0.05 and 0.9; therefore, we studied the SNR under CR of 0.5 and 0.9. Figure 5.10(a) demonstrates that when the ChS increases, the RER also increases. Under the same ChS, when SNR equals 3, the reconstructed error are larger than that of SNR equaling 4 to 6, but the RER differs little when SNR equals 4 to 6. The same results appear in Figure 5.10(b), in which CR equals 0.9. Comparing Figure 5.10(a) and 5.10(b), it shows that the results are almost the same. The simulation results prove that SNR has influence on the reconstruction error, and using high SNR has better reconstruction performance for each value of ChS, but when the SNR increases to 4 or higher, the reconstruction error does not decrease. In addition, SNR has no obvious influence on the CR.

Power consumption is a major concern in the design of the multichannel system. Figure 5.11 shows the relationship between the scan rate and power consumption. It can be found that the power consumption and the scan rate have an approximately linear relationship. In our design, suppose the sampling frequency is 25 kHz for each channel, and that we want to design a 256-

channel system, if we want to acquire the best reconstruction performance, that is, the minimum RER, we need to use a scan rate of 400 kHz for both modes. The power consumption is around 800 μW in this case, which may be too large for an implantable neural recording system, but it turns out that using the maximum scan rate is not necessary. When we reduce the scan rate to 200 kHz, 100 kHz and 50 kHz, the power consumption correspondingly reduces to be around 400 μW, 200 μW and 100 μW. Meanwhile, as discussed above, the RER also increases when the scan rate is reduced. Therefore, the designer needs to make a compromise between the power consumption and the reconstruction performance.

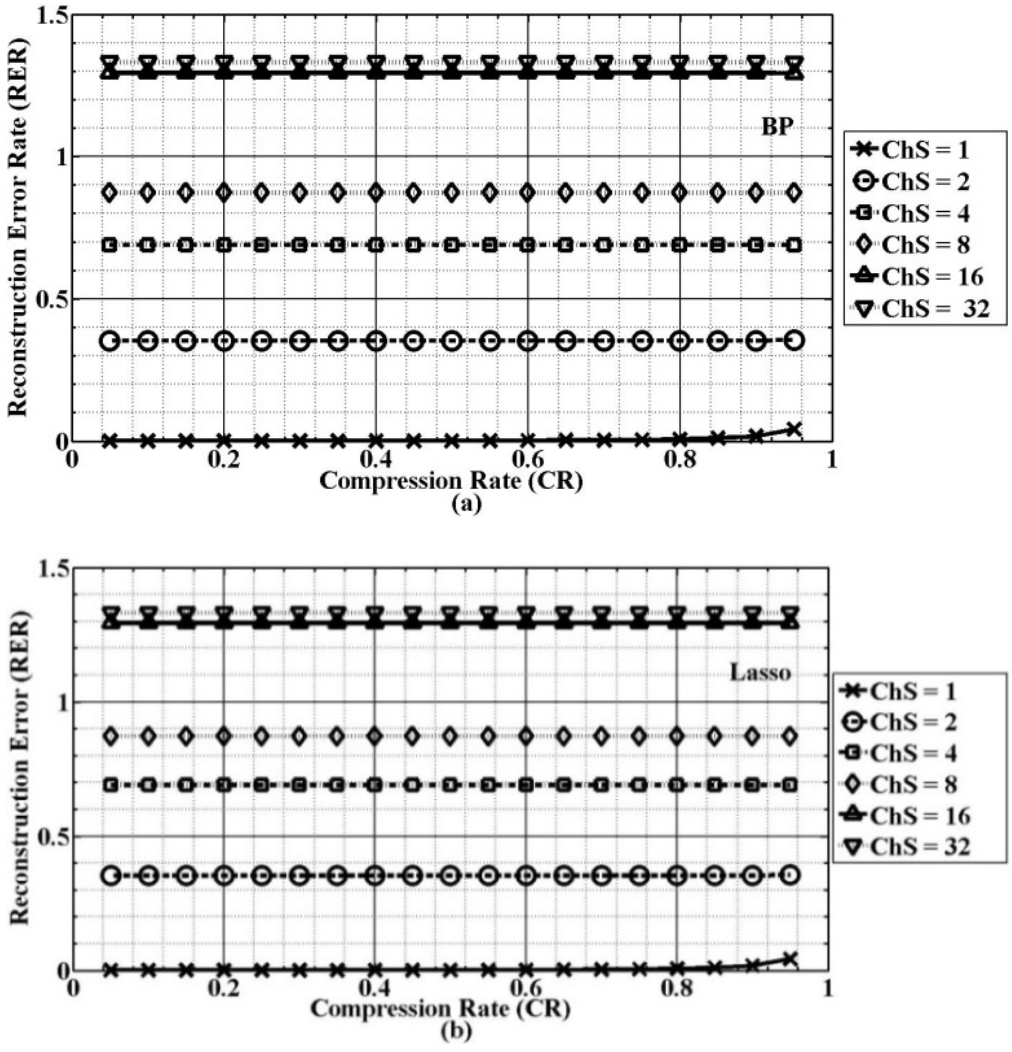


Figure 5.9 Relationship between the compression rate and the reconstruction error for the multichannel using: (a) BP algorithm, (b) Lasso algorithm

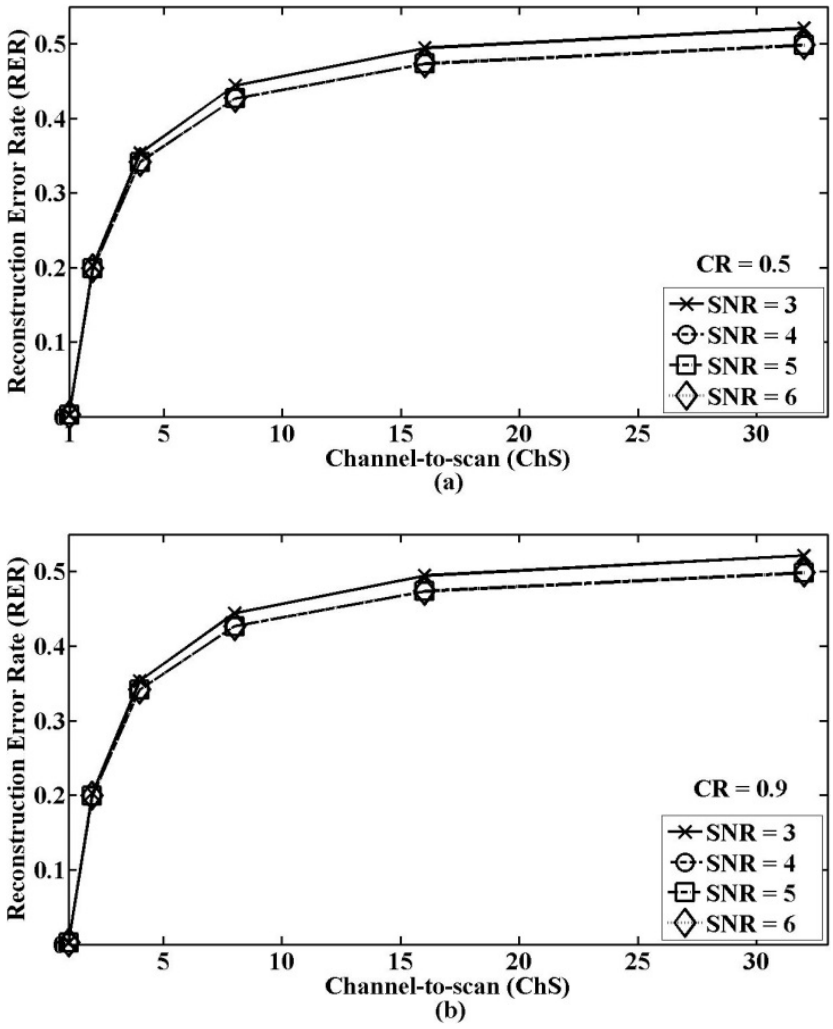


Figure 5.10 Relation among channel-to-scan, SNR and reconstruction error rate for the multichannel processing using: (a) compression rate = 0.5, (b) compression rate = 0.9

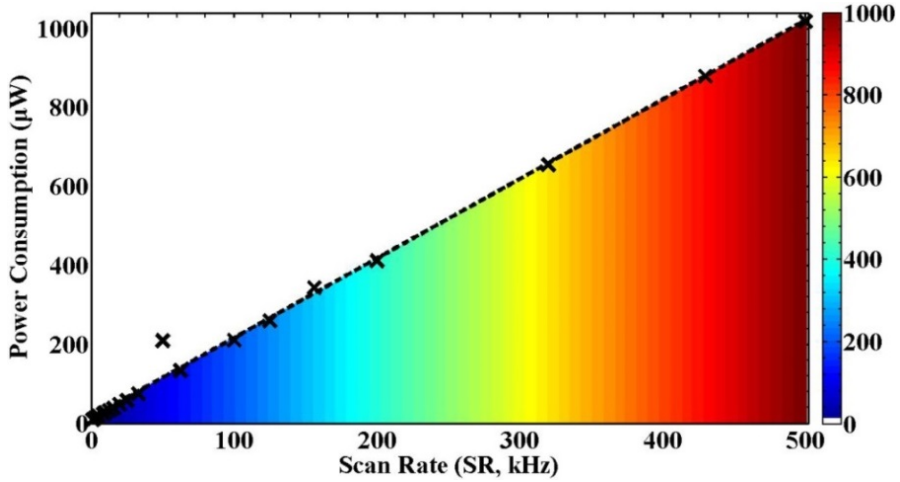
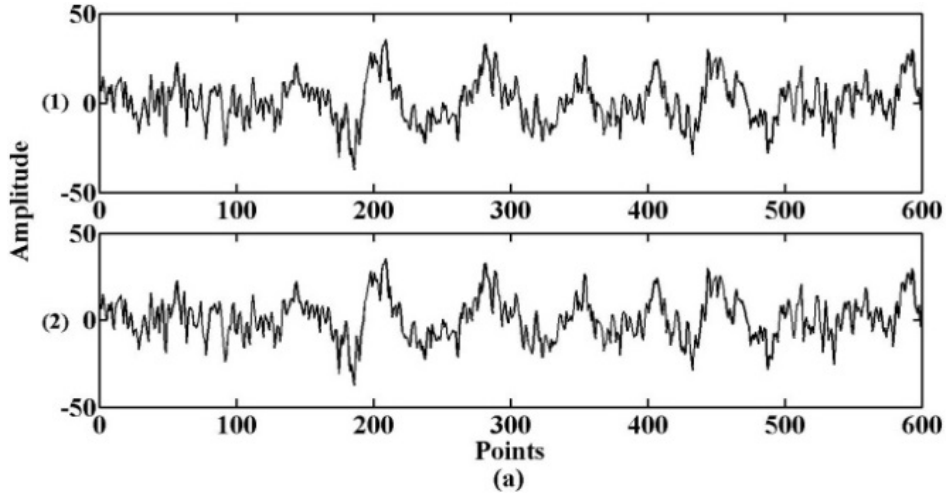


Figure 5.11 Relation between the scan rate and the power consumption

Finally, we discuss how to choose the number of the blocks and corresponding channels per block. The configuration is based on the demands of the research. Suppose the distances between two electrodes d_1 and between two blocks d_2 ($d_1 \neq d_2$) are determined; if users want to record the signals from adjacent channels, they should choose more channels per block; if users want more information from the channels within a relatively large distance, they should choose more blocks. Moreover, the design of the channels also depends on the size of the device. In our design, we think both aspects are important, so we designed a $2^4 \times 2^4$ array to process the signals.

5.5.3 The Reconstruction under Multichannel Operation

The reconstruction of the original signals under different ChS and an example of a 16-channel (or 16 blocks) reconstruction is described. First of all, the reconstruction of the signals under different ChS is discussed. Figures 5.12(a)-(1), (b)-(1), (c)-(1), (d)-(1) show the original signals and Figures 5.12(a)-(2), (b)-(2), (c)-(2), (d)-(2) are their corresponding reconstructions. The comparison shows that when the ChS equals 1, the reconstructed signal has the best resolution, and it is nearly identical to the original signal. With increasing ChS, the resolution of the reconstructed signal is reduced. When ChS equals 4, the RER is around 0.7, but the reconstructed signal still has good resolution and keeps some details. When ChS equals 8, the reconstructed signal begins to lose details but keeps the morphology of the original signal. Therefore, when the design needs more details from a signal, choosing a small ChS (1,2,4) is a good option; when the details of the signal are not important and the shape of the signal is sufficient, using a larger ChS (8) is better.



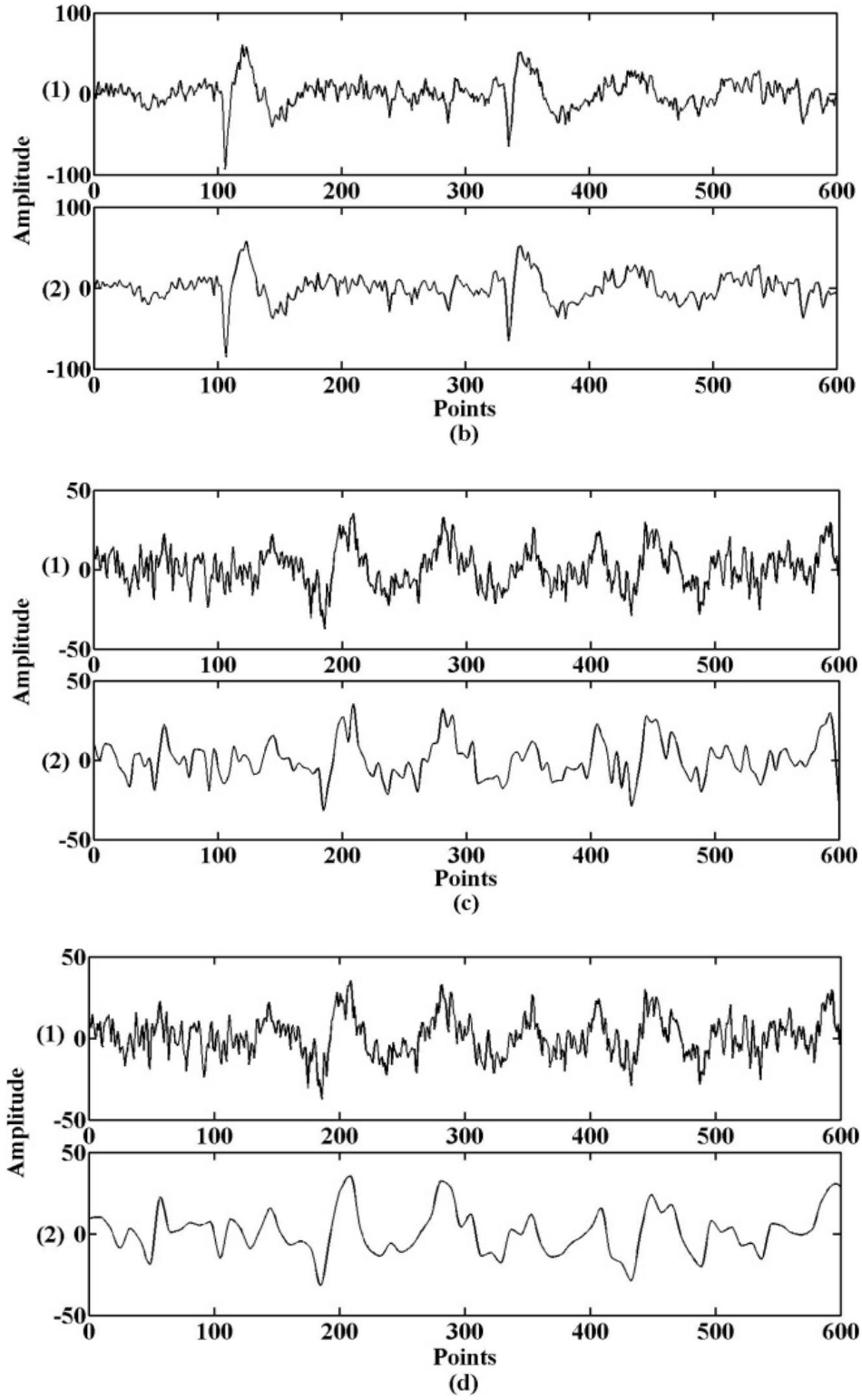
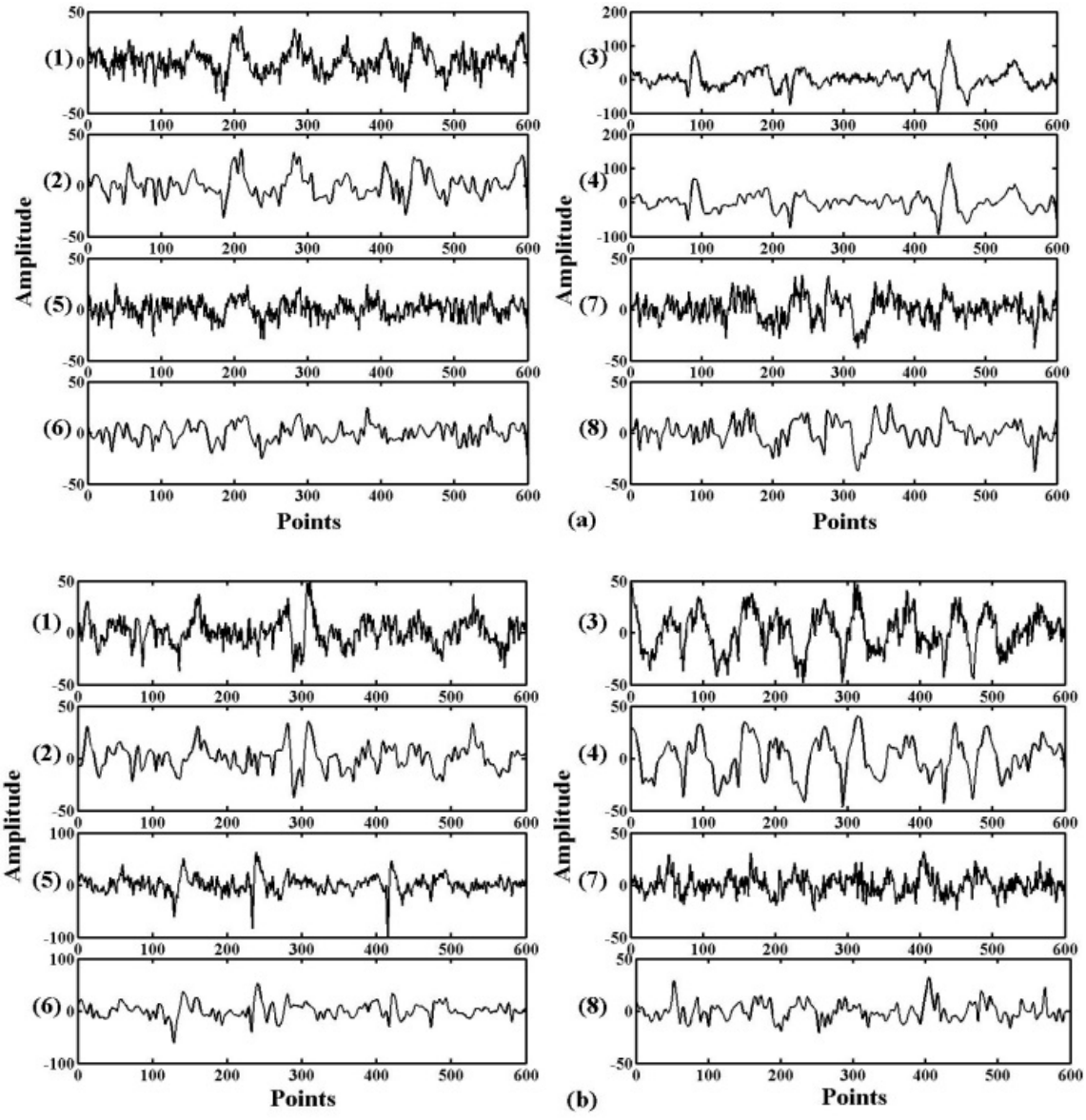


Figure 5.12 The comparison between original signals and their reconstructed signals under different ChS: (a) ChS = 1, (b) ChS = 2, (c) ChS = 4, (d) ChS = 8

The ChS can be increased in two ways: reducing the scan rate control parameter, which means acquiring fewer samples, or keeping the maximum scan rate control parameter and reducing the sampling rate. A lower sampling rate reduces the power consumption of the analog-to-digital converter of the system, which is appropriate for a system that does not need to record many details of signals. If we want to design a frequency-changeable system, it is better to use a higher sampling frequency.



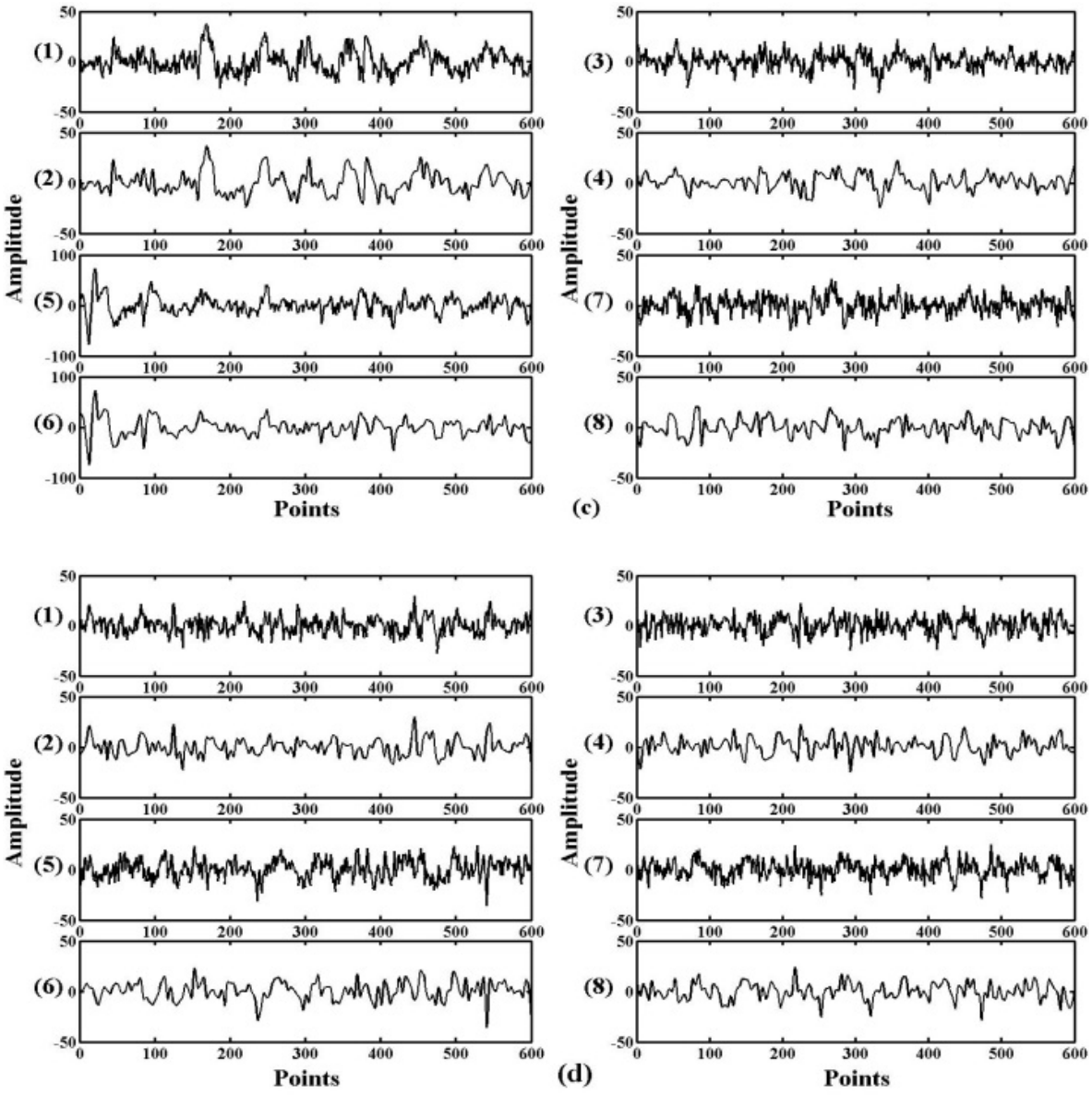


Figure 5.13 An example of the original signals and their reconstructed signals from 16 channels: (a) channels 1-4, (b) channels 5-8, (c) channels 9-12, (d) channels 13-16

A comparison of the original signals and their reconstructed signals for 16 blocks is given (Figure 5.13). In our design, we chose a ChS equaling 4 and a sampling rate of every channel of 25 kHz to process the 256-channel system; all the reconstructed signals and their original signals are illustrated. Figures 5.9 and 5.13 show that the reconstructed signals are very similar to the original signals from these 16 channels, even where the compression rate is around 90%.

5.5.4 Other Important Results

Figure 5.14(a) shows the output format of the system. There are two output ports: one is for the output of the sensing matrix and detection, and the other is for the compressed data. The first port outputs the indicator for the start of the sensing matrix (SSM, one-bit), the indicator for the start of the detection (ID, 1-bit), the matrix results (MR, 5-bit) and the detection results (DR, 2-bit). There are two formats for this output. The first format output (FO) is 9 bits (including the start bit and stop bit), when there are no detection results (ID = 0). When detection begins (ID = 1), the output is 11 bits, which is transmitted as the second format (SO). The second port outputs an indication of the start of the compressed signal (SCS) and the compressed data (CD). The format of the output from this port is called compressed data output format (CO). After every 1000 sampled points, compressed signals are sent out. Each compressed data unit has 23 bits. When the first port outputs the data, the second port stops sending data out, and vice versa. The timing diagrams of the three formats are shown in Figures 5.14(b) and (c).

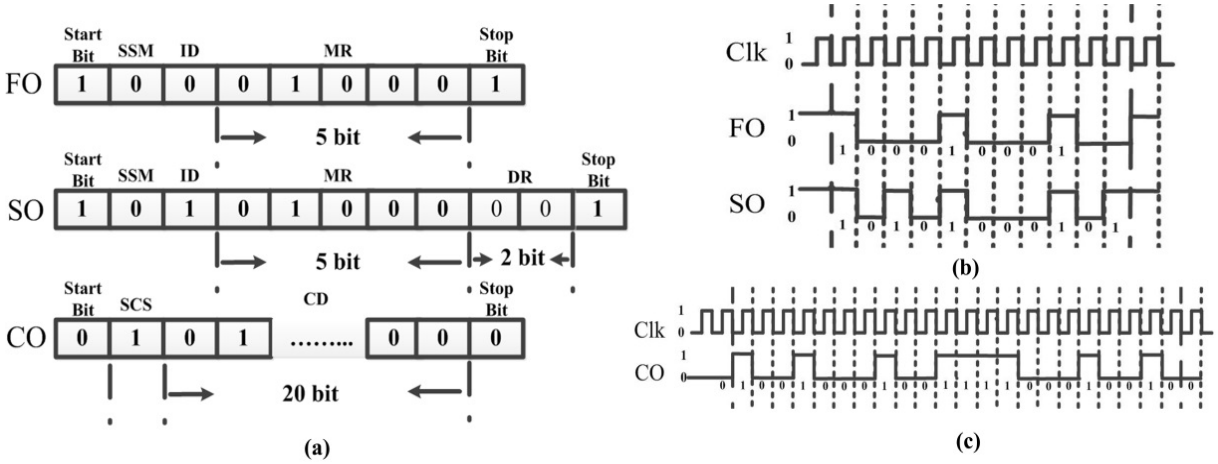


Figure 5.14 The output of the digital circuit: (a) format of the outputs, (b) timing diagram of the FO and SO, (c) timing diagram of the CO

The post-layout diagram and the real tested results are given in Figure 5.15. It can be seen that the core area of the custom chip is around 0.49 mm^2. Figure 5.16(a) shows the FPGA-based simulation. We randomly generated one period of the data, imitating the multichannel recording, to test the system. Figure 5.16(b) shows the first format output (FO) from the test board, Figure 5.16(c) shows the second format output (SO) from the test board, and Figure 5.16(d) shows the output format of the compressed signals (CO) from the FPGA board. The power consumption of

the system (256-channel, ChS equals 4) is around 200 µW (12.5 µW/channel) and the area is around 0.49 mm^2 (0.03 mm^2/channel), as estimated by Synopsys and Cadence using IBM CMOS130.

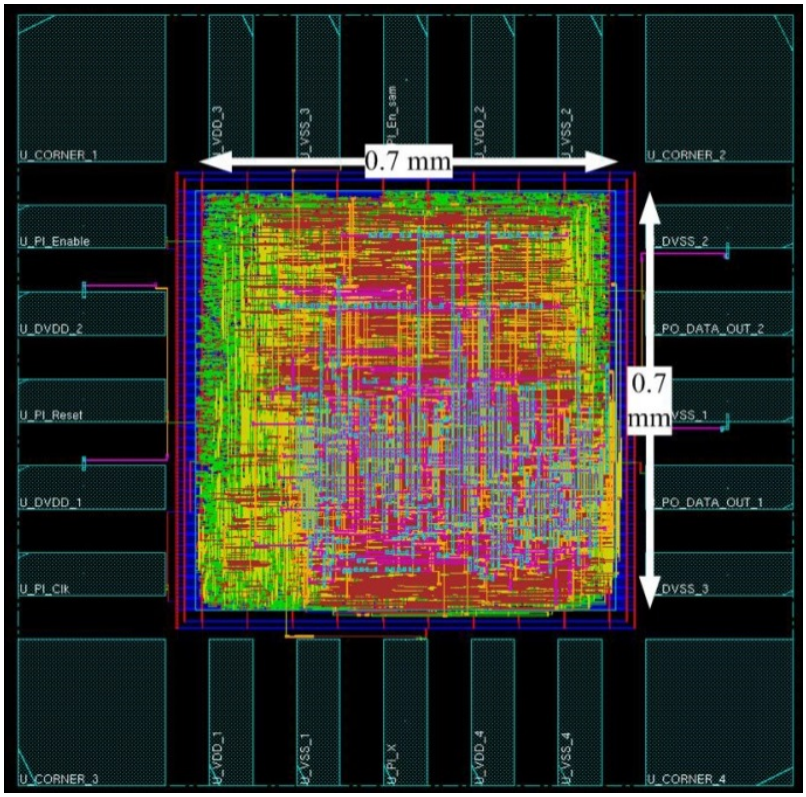


Figure 5.15 Post-layout of the proposed 256-channel digital neural signal processing system

Finally, we compare our results with those from related publications in Table 5.2. Our work is based on a digital circuit design and the MDC matrix-based CS technique. Table 5.2 shows our work has relatively low power consumption and a small area.

Table 5.2 Comparison of proposed MDC-based digital neural signal processing system with similar existing systems

Reference	**[225]**	**[226]**	**[227]**	**[166]**	**[167]**	**[168]**	**This Work**
Technology (µm CMOS)	0.35	0.5	0.18	0.5	0.5	0.065	0.13
Supply voltage (V)	1.5	3.3	1.8	–	3	0.27	1.2

Reference	[225]	[226]	[227]	[166]	[167]	[168]	This Work
Compression method	Spike Waveform	Spike detection	Spike detection	Spike detection	Spike detection	Spike sorting	Digital CS
Number of channels	16	100	16	32	32	16	256
Area per channel (mm^2/channel)	-	< 0.16	>0.0475	0.18	0.12	0.07	0.03*
Power consumption per channel (µW/channel)	269	27	>96	95	75	4.68	12.5
Sampling rate per channel(kS/s)	1250	15	30	25	20	-	25

* This includes the core area only.

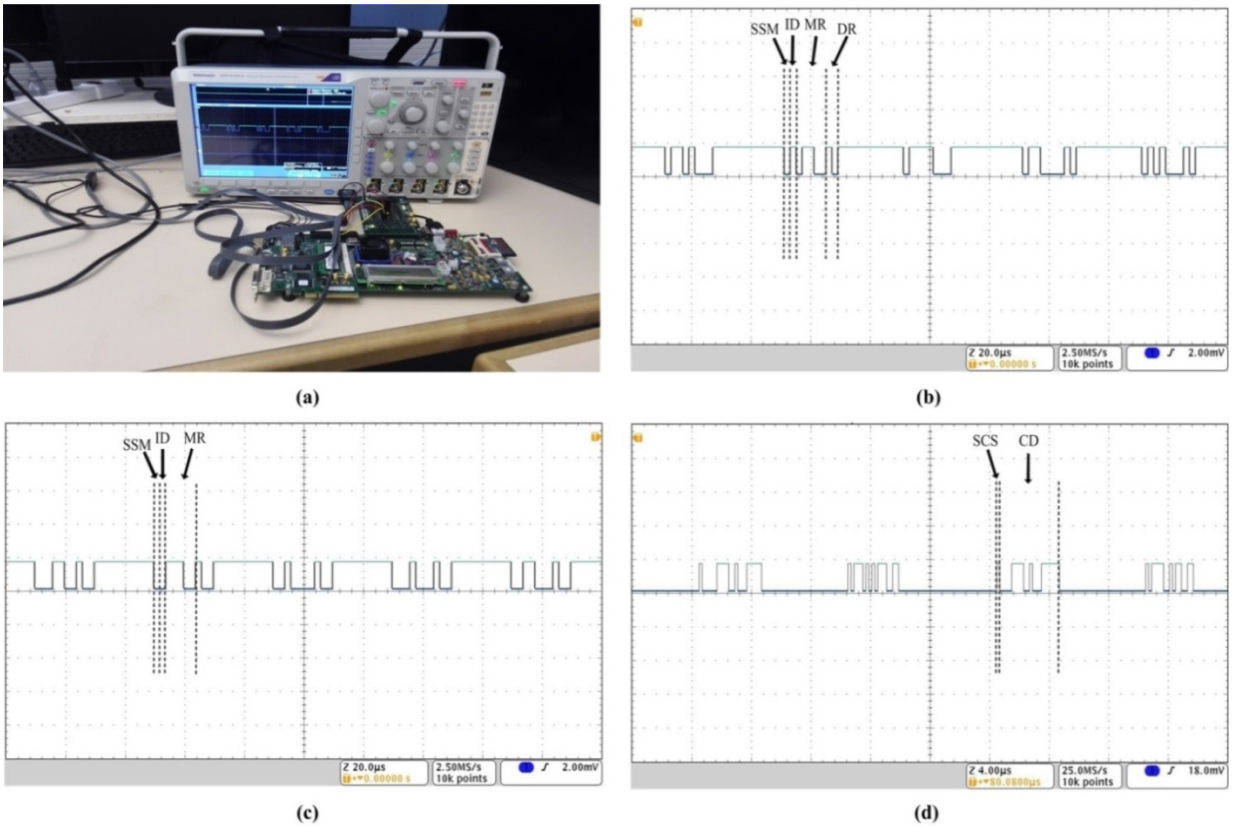


Figure 5.16 The FPGA-based simulation: (a) the picture of the FPGA-based test system, (b) FO from the FPGA board, (c) SO from the FPGA board, (d) CO from the FPGA board

5.6 Conclusions

In this article, we put forward a multichannel digital neural signal processing system using an MDC matrix. We introduced the construction of the MDC matrix and we discussed the construction of a single-channel signal processing system. The single-channel system includes two building blocks: the spike detection block and the data compression block. We chose the RMS method to detect the spikes and applied the MDC matrix to compress neural signals. When using the MDC matrix to compress the signal, the distance between the current data point and the core data point is an important parameter. We evaluated the relationship between the distance and the reconstruction error with two reconstruction algorithms. We also explained the relationship between the compression rate and the distance, and we found that choosing 4 or 5 for the distance σ is appropriate. Additionally, it can be proved that the original signal can be recovered by both BP and Lasso algorithms. The construction of the multichannel system was detailed, where a scan was applied to process signals, and both the scan direction and the scan rate were analyzed. The scan rate has a tight relationship with the power consumption and reconstruction performance. The lower the ChS is set, the better the reconstruction performance, while also demanding greater power consumption. Based on the discussion, we put forward a 256-channel ($2^4 \times 2^4$) signal processing system with a ChS equaling 4. The power consumption of this system is about 12.5 μW/channel and the area is around 0.03 mm^2/channel, and compression rate is around 90% while the reconstructed signals keep most of the details of the original signals. Finally, an example of 16-channel original signals and their corresponding reconstructed signals were provided. The post-layout diagram and FGPA-based real signal tests were given, and a comparison with other similar works was given to highlight the importance of our proposed system. From the simulation results and comparison, we found that our system has not only large compression rate and good reconstruction accuracy but also relatively low power consumption and a small area.

CHAPTER 6 GENERAL DISCUSSION

We describe, in this thesis, systematic and detailed methodologies into the design of neural signal processing for neural recording devices. In the early days, neural recording device designers focused on acquiring and transmitting neural signals, but the needed high and increasing data requirements of modern neural recording devices mean that only designing the signal acquisition or transmission components is insufficient, and integrating a high-performance neural signal processing system becomes more and more important.

A neural signal processing device should have high processing performance; for example, the spike detection block should have a low false positive rate and high true positive rate; the signal compression system needs to be designed with a high compression rate and low reconstruction error. Also, because the proposed neural processing techniques are designed for implantable neural recording devices, they should have low complexity architecture, which eases the circuit design. In this thesis, several new techniques for neural signals processing were developed and tested; these techniques have relatively simple structures that are easily implemented and present high accuracy for their specific usage.

For signal processing of neural recording interfaces, there are principally two strategies: signal reduction and compression, and both techniques were investigated in this thesis. In chapter 3, we focused on the signal compression strategy, and proposed a new method to construct the sensing matrix based on the compressed sensing (CS) technique, which can be used to effectively compress neural signals. In chapter 4, we mainly discussed the signal reduction strategy. In this chapter, we designed an automatic template matching system to make the spike detection, which can be used to remove the noise of neural signals. In addition, in chapter 5, based on both strategies, we designed a digital signal processing system, which includes the amplitude-based spike detection block and CS-based signal compression block. The signal compression block contains a subblock generating the MDC matrix which is discussed in chapter 3.

In chapter 2, we review several state-of the-art works, and we also compare our works with these state-of-the-art works in chapter 2-5. In Table 6.1, we continue to discuss the comparison between our work and the reviewed state-of-the-arts works, and summarize the contribution of our research.

Table 6.1 Discussion of contribution of our work comparing with the state-of-the-arts works

	Summary of comparison of state-of-the-arts works	**Summary of the contribution of our research**
First work	**Table 2.4**	**Tables 2.4 and 3.3**
	1. Only for sparse signals 2. Have high complexity 3. The processing performance can be further improved	1. Our proposed sensing matrix can compress low-sparsity and non-sparse signals 2. The proposed sensing matrix has low complexity 3. The proposed sensing matrix can compress sparse and non-sparse with a relatively large compression rate and a small reconstruction error
Second work	**Table 2.7**	**Tables 2.7 and 4.1**
	1. Several systems are not automatic template matching systems 2. Have high complexity 3. Detection accuracy can be improved	1. Our system has low complexity and relatively good detection performance 2. Our system is an automatic template matching-based system, the templates need not be foreknown.

Table 6.1 Discussion of contribution of our work comparing with the state-of-the-arts works (cont'd)

	Summary of comparison of state-of-the-arts works	**Summary of the contribution of our research**
Third work	**Table 2.5**	**Tables 2.5 and 5.2**
	1. Cannot make the spike detection and signal compression in the same processor 2. CS-based compressor only compress sparse signals 3. Power consumption and area still can be reduced	1. Our processing system can compress low-sparsity and non-sparse signals 2. The implemented system include spike detector and CS-based signal compressor 3. Our system has a relatively low power consumption, small area and good processing performance

More specifically, concerning our first work in chapter 3, from Tables 2.4 and 3.3, it can be found that for compared sensing matrices, they only compress sparse signals, but lots of signals are not sparse even applying approximation or changing the bases. In our research, we found that some non-sparse signals which contain identical points can be also compressed, and we proposed a method to construct the MDC sensing matrix using the concept of similarity. Besides, our proposed sensing matrix can compress the sparse and non-sparse signals with a large compression rate and a small reconstruction error, which is better than compared systems; for example, the MDC matrix can compress non-sparse and sparse neural signals (degree of sparsity equaling 0 and 50%) with a compression rate equaling 98%, and reconstruction errors are both lower than 0.2 when using basis pursuit reconstruction algorithm. Finally, our sensing matrix is a deterministic sensing matrix and also composed of zeros and ones, comparing with the similar random or deterministic sensing matrices, such as digital wavelet transform-based sensing matrix, chirp sensing codes matrix, Bose-Chaudhuri-Hocquenghem matrix, etc., our proposed sensing matrix has low complexity, which is suitable for hardware implementation.

Second, regarding our second work in chapter 4, from Tables 2.7 and 4.1, it can be found that our

proposed Bayesian inference-based automatic template matching spike detection and classification system has a simple structure and also the least calculation in the comparison tables, which is suitable for the hardware design. Besides, this system is an automatic template generation system, and the templates need not to be given in advance, which is better than several compared template matching-based spike detection systems. Finally, comparing with several amplitude-based, energy-based and template matching-based system, our system has relatively high detection accuracy.

Third, we compare several state-of-the-arts neural signal processing systems and also compare our third work with them in Tables 2.5 and 5.2. Firstly, our proposed system is based on our proposed MDC matrix; therefore, it can be used to compress sparse and non-sparse neural signals, which are better than the other compared systems. Besides, our system includes spike detector and CS-based signal compressor. Comparing with the other systems, our system is the only system providing both functions of spike detection and signal compression. Finally, our system supports single-channel and multichannel processing, and also comparing with the compared systems, it has relatively low power consumption and a small area.

For each work, first, we studied the CS technique, a new signal processing method for compression of neural signals. Whether or not neural signals are sparse is still in dispute. In our research, we found that neural signals are not sparse in the time domain, so directly applying CS technique is not appropriate. Fortunately, we found that neural signals have a lot of similar points. Compared with traditional CS technique based on the sparsity, the proposed method can be innovatively applied for signal compression. We investigated the use of the restricted isometry property of the MDC matrix for compression. The simulation results show that with the MDC matrix, the compression rate of the signal can reach 90%, and the reconstruction error is lower than 10% using the Basis Pursuit reconstruction algorithm. Also, the MDC matrix can be composed of zeros and ones, which has low complexity. A comparison between the proposed method and the other CS-based compression ones revealed that the proposed system can compress signals with a large compression rate and small reconstruction error, and can be used for the compression of non-sparse signals. Therefore, the MDC matrix is a good candidate for implementation in a neural recording system.

On the other hand, signal reduction methods remove useless information (noise) from signals and

only keep the spikes that neurons generate. These spikes are usually composite signals generated by different neurons, whereas the spikes from single-unit neuron are usually needed for various research purposes. Traditional amplitude-based and energy-based spike detection methods do not have enough detection accuracy for use on low-SNR signals. Furthermore, neither methods can be directly used for spike classification. The template matching method has good detection accuracy for low-SNR signals, and can be directly used for spike classification, but this method is complicated and needs foreknown templates. We proposed a system using a Bayesian Inference template matching method to perform the spike detection and sorting. Compared with the amplitude-based or energy-based spike detection methods, the BBTM method has high detection accuracy for the low SNR signal and can perform spike classification. Compared with some other template matching methods, BBTM has a simple structure, high detection and classification accuracy, and the templates need not be foreknown.

Taking into consideration detailed study on signal reduction and compression techniques, the power consumption limits, the small area, and other important parameters, a neural recording module including spike detection and signal compression was proposed. Based on this module, single-channel and multichannel signal processing architectures were investigated and validated. The simulation results showed that the proposed module has a relatively small area and low power consumption, compared with several existing neural signal processing ones. In addition, the proposed module was tested and verified through an FPGA testing board (Virtex-6 FPGA ML605). The proposed digital signal processing module not only performs both spike detection and compression, but has relatively low power consumption and a small area.

In summary, this thesis involves the key points related to neural signal processing activities. The research covered several neural signal processing techniques, and a digital neural processing system was proposed. We proposed two innovative methods for the neural signal processing based on the CS and template matching techniques, and we provided methods for the construction of new and efficient neural processing devices. Also, the corresponding circuit implementation is fulfilled. Through our research, the proposed methods cannot only be applied in the design of the neural recording device, but also the signal processing for other biomedical signals, such as MEG, ECoG, or the image , video or speech processing .

CHAPTER 7 CONCLUSION AND RECOMMENDATIONS

7.1 Conclusion

We described in this thesis several methods intended for neural signal reduction and compression using template matching and CS techniques, which culminated in a new and highly efficient digital neural signal processing system.

First of all, the minimum Euclidean or MDC sensing matrix for neural signal compression was generated, and its restricted isometry property was proved. Proving the restricted isometry property of the MDC matrix required the satisfaction of two prerequisites: that $(k-M)/N \to 0$ (k is the sparsity of the signal, M and N are the numbers of rows and columns of the sensing matrix), and that the clustering must be more even and $\mathrm{I}_{\max}(\mathrm{Set}(C)) \leq N/M$. Also, several reconstruction algorithms for the reconstruction of original signals were evaluated. The simulation results confirmed that BP and Lasso algorithms are useful reconstruction algorithms. The influence of the sampling rate and length of data on the compression and reconstruction was also examined, and it was found that a UMDC matrix composed of zeros and ones is appropriate for the circuit implementation.

For signal reduction, we proposed a Bayesian inference-based template matching method which can automatically generate templates. The BBTM method has better detection accuracy than the amplitude-based or energy-based spike detection methods when the templates are known. When the templates are unknown, we used the maximum minimum spread sorting method to generate the templates; the true positive rate can reach up to 0.95 with a false positive rate of 0.05. The BBTM method also enables spike classification. We used correlation and Euclidean distance to estimate the difference between the templates and processed neural signals, and the thresholds of both estimations can be set as 0.8 and 0.5 respectively. The clustering accuracy is around 1 when false positive rate equals 0.1 for three-neuron composite signals. The BBTM method has low computation complexity, requiring only around 1.2 ms for spike detection and clustering of each spike.

Finally, we proposed a digital neural signal processing system including spike detection and compression building blocks. For the single-channel design, the root mean square method and MDC matrix were chosen for spike detection and signal compression. We evaluated the

relationship between the distance and the reconstruction error using two reconstruction algorithms, and also the relationship between the distance and compression rate. Choosing 4 or 5 for the distance σ proved to be appropriate. In contrast, the construction of the multichannel system is detailed, in that a scan is applied to process signals and both the scan direction and rate must be analyzed. The scan rate has a tight relationship with the power consumption and reconstruction performance. The lower the channel-to-scan parameter is set, the better the reconstruction performance, but the higher the power consumption. As found by one of the most advanced simulation tools, Synopsys, the power consumption and area of the proposed 256-channel system are 12.5 μW/channel and 0.03 mm^2/channel respectively.

Our research demonstrated that the MDC matrix and BBTM methods have good compression and detection performance, and the proposed overall neural signal processing module has small area and low power consumption comparing with existing modules while maintaining good compression performance.

7.2 Recommendation for Future Work

We described, in this thesis, a method that uses the similarity of spikes to compress signals. Sparsity can be regarded as a special form of similarity, as all data are zeros. Based on this idea, more sensing matrices can be designed and evaluated for their signal compression performance. We recommend to conduct further investigations on the properties of the sensing matrix and develop more mathematical theories on the construction of the sensing matrix and the reconstruction of the original signals based on similarity.

The BBTM method used in this thesis is an automatic template generation method. We used the K-mean clustering method for final clustering, and adapted an Osort algorithm to determine K. The accuracy of the determination of K for this algorithm can be further improved, and is an obvious avenue for more research.

The systems based on the BBTM method and the MDC matrix both need to be optimized. In particular, the power consumption of the digital circuit based on the BBTM method should be reduced. Similarly, the performance of the circuit based on the MDC matrix can be enhanced. In future work, we recommend to concentrate on designing a circuit with even lower power consumption using both the BBTM method and the MDC matrix.

Finally, the front-end circuit and wireless transmitter should be added into the proposed neural signal processing module, and a system-on-chip neural recording device should be fabricated, and a test *in vivo* must be performed. In future work, the transmitter needs to be implemented, and the design of the custom neural recording device will be researched.

REFERENCES

[1] S. Motamedi-Fakhr, M. Moshrefi-Torbati, M. Hill, *et al.*, "Signal processing techniques applied to human sleep EEG signals—A review," *Biomedical Signal Processing and Control*, vol. 10, pp. 21-33, Mar. 2014.

[2] A. Caria, R. Sitaram, and N. Birbaumer, "Real-Time fMRI a tool for local brain regulation," *The Neuroscientist*, vol. 18, pp. 487-501, 2012.

[3] D. Mantini, S. D. Penna, L. Marzetti, *et al.*, "A signal-processing pipeline for magnetoencephalography resting-state networks," *Brain connectivity*, vol. 1, pp. 49-59, 2011.

[4] K. Muller, T. Adali, K. Fukumizu, *et al.*, "Special Issue on Advances in Kernel-Based Learning for Signal Processing," *Signal Processing Magazine, IEEE*, vol. 30, pp. 14-15, 2013.

[5] Z. Saad Zaghloul and M. Bayoumi, "Adaptive neural matching online spike sorting VLSI chip design for wireless BCI implants," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, 2015, pp. 977-981.

[6] T. Gürel and C. Mehring, "Unsupervised Adaptation of Brain-Machine Interface Decoders," *Frontiers in Neuroscience*, vol. 6, p. 164, Nov. 2012.

[7] Y. X. Yang and M. M. Shanechi, "An adaptive brain-machine interface algorithm for control of burst suppression in medical coma," in *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, 2014, pp. 1638-1641.

[8] H. Mamaghanian, N. Khaled, D. Atienza, *et al.*, "Compressed Sensing for Real-Time Energy-Efficient ECG Compression on Wireless Body Sensor Nodes," *Biomedical Engineering, IEEE Transactions on*, vol. 58, pp. 2456-2466, 2011.

[9] S. Aviyente, "Compressed Sensing Framework for EEG Compression," in *Statistical Signal Processing, 2007. SSP '07. IEEE/SP 14th Workshop on*, 2007, pp. 181-184.

[10] M. A. Shaeri and A. M. Sodagar, "A Method for Compression of Intra-Cortically-Recorded Neural Signals Dedicated to Implantable Brain-Machine Interfaces," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 23, pp. 485-497, 2015.

[11] J. Zhang, Y. M. Suo, S. Mitra, *et al.*, "An Efficient and Compact Compressed Sensing Microsystem for Implantable Neural Recordings," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 8, pp. 485-496, 2014.

[12] D. E. Bellasi, R. Rovatti, L. Benini, *et al.*, "A Low-Power Architecture for Punctured Compressed Sensing and Estimation in Wireless Sensor-Nodes," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 62, pp. 1296-1305, 2015.

[13] R. M. Rangayyan, " Introduction to Biomedical Signals" in *Biomedical signal analysis: a case-study approach*. New York: John Wiley & Sons, Inc., 2012, pp. 1 - 59

[14] L. F. Nicolas-Alonso and J. Gomez-Gil, "Brain Computer Interfaces, a Review," *Sensors*, vol. 12, pp. 1211-1279, 2012.

[15] J. J. Shih, D. J. Krusienski, and J. R. Wolpaw, "Brain-Computer Interfaces in Medicine," *Mayo Clinic Proceedings*, vol. 87, pp. 268-279, Mar. 2012.

[16] M. A. Lebedev and M. A. L. Nicolelis, "Brain–machine interfaces: past, present and future," *Trends in Neurosciences*, vol. 29, pp. 536-546, Sept. 2006.

[17] R. Salmelin, M. Hámáaláinen, M. Kajola, *et al.*, "Functional Segregation of Movement-Related Rhythmic Activity in the Human Brain," *NeuroImage*, vol. 2, pp. 237-243, Dec. 1995.

[18] J. Freeman. "Cognitive Consonance: Neuroimaging: EEG, MRI, fMRI, MEG, PET and TMS." Internet: http://cognitiveconsonance.info/2014/01/13/updated-neuroimaging-eeg-mri-fmri-meg-pet-and-tms/ [2015].

[19] E. Flynn, "Magnetic Relaxometry: A Comparison to Magnetoencephalography," in *Magnetoencephalography*, S. Supek and C. J. Aine, Eds. Berlin Heidelberg: Springer, 2014, pp. 979-991.

[20] M. Sawan, M. T. Salam, J. Le Lan, *et al.*, "Wireless Recording Systems: From Noninvasive EEG-NIRS to Invasive EEG Devices," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 7, pp. 186-195, 2013.

[21] D. K. Nguyen, J. Tremblay, P. Pouliot, *et al.*, "Non-invasive continuous EEG-fNIRS recording of temporal lobe seizures," *Epilepsy Research*, vol. 99, pp. 112-126, Mar. 2012.

[22] G. Buzsáki, C. A. Anastassiou, and C. Koch, "The origin of extracellular fields and currents — EEG, ECoG, LFP and spikes," *Nature Reviews Neuroscience*, vol. 13, pp. 407-420, June 2012.

[23] T. Ball, M. Kern, I. Mutschler, *et al.*, "Signal quality of simultaneously recorded invasive and non-invasive EEG," *NeuroImage*, vol. 46, pp. 708-716, July 2009.

[24] N. K. Logothetis, J. Pauls, M. Augath, *et al.*, "Neurophysiological investigation of the basis of the fMRI signal," *Nature*, vol. 412, pp. 150-157, July 2001.

[25] P. Pouliot, J. Tremblay, M. Robert, *et al.*, "Nonlinear hemodynamic responses in human epilepsy: A multimodal analysis with fNIRS-EEG and fMRI-EEG," *Journal of Neuroscience Methods*, vol. 204, pp. 326-340, Mar. 2012.

[26] C. J. Price, "A review and synthesis of the first 20 years of PET and fMRI studies of heard speech, spoken language and reading," *NeuroImage*, vol. 62, pp. 816-847, Aug. 2012.

[27] P. Molenberghs, R. Cunnington, and J. B. Mattingley, "Brain regions with mirror properties: A meta-analysis of 125 human fMRI studies," *Neuroscience & Biobehavioral Reviews*, vol. 36, pp. 341-349, Jan. 2012.

[28] G. de Zubicaray, K. Johnson, D. Howard, *et al.*, "A perfusion fMRI investigation of thematic and categorical context effects in the spoken production of object names," *Cortex*, vol. 54, pp. 135-149, May 2014.

[29] R. R. Harrison, "The Design of Integrated Circuits to Observe Brain Activity," *Proceedings of the IEEE*, vol. 96, pp. 1203-1216, 2008.

[30] R. M. a. J. E. R. Dowben, "A Metal-Filled Microelectrode," *Science* vol. 118(3053), pp. 22-24, 1953.

[31] J. D. Green, "A Simple Microelectrode for recording from the Central Nervous System," *Nature*, vol. 182(4640), pp. 962-962, 1958.

[32] H. Jamille and A. David, "Silicon microelectrodes for extracellular recording," in *Handbook of Neuroprosthetic Methods*, Florida: CRC Press, 2002, pp. 163 - 191.

[33] E. a. J. E. A. Marg, "Indwelling Multiple Micro-Electrodes in the Brain," *Electroencephalography and Clinical Neurophysiology*, pp. 277-280, 1967.

[34] Polystim. "Multichannel implantable neural signal acquisition system." Internet: http://www.polystim.org/?page=axes-projets.php [2015].

[35] B. Gosselin and M. Sawan, "A low-power integrated neural interface with digital spike detection and extraction," *Analog Integrated Circuits and Signal Processing*, vol. 64, pp. 3-11, July 2010.

[36] R. Chebli and M. Sawan, "Low noise and high CMRR front-end amplifier dedicated to portable EEG acquisition system," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, 2013, pp. 2523-2526.

[37] B. Gosselin, M. Sawan, and E. Kerherve, "Linear-Phase Delay Filters for Ultra-Low-Power Signal Processing in Neural Recording Implants," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 4, pp. 171-180, 2010.

[38] L. Liu, X. Zou, W. L. Goh, *et al.*, 800 nW 43 nV/ √ Hz neural recording amplifier with enhanced noise efficiency factor. *Electronics Letters*, vol.48(9), pp. 479-480, 2012.

[39] K. A. Ng and X. Yong Ping, "A Compact, Low Input Capacitance Neural Recording Amplifier," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 7, pp. 610-620, 2013.

[40] K. Abdelhalim, L. Kokarovtseva, J. L. Perez Velazquez, *et al.*, "915-MHz FSK/OOK Wireless Neural Recording SoC With 64 Mixed-Signal FIR Filters," *Solid-State Circuits, IEEE Journal of*, vol. 48, pp. 2478-2493, 2013.

[41] X. D. Zou, L. Liu, J. H. Cheong, *et al.*, "A 100-Channel 1-mW Implantable Neural Recording IC," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 60, pp. 2584-2596, 2013.

[42] M. Judy, A. M. Sodagar, R. Lotfi, *et al.*, "Nonlinear Signal-Specific ADC for Efficient Neural Recording in Brain-Machine Interfaces," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 8, pp. 371-381, 2014.

[43] V. Chaturvedi, T. Anand, and B. Amrutur, "An 8-to-1 bit 1-MS/s SAR ADC With VGA and Integrated Data Compression for Neural Recording," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, pp. 2034-2044, 2013.

[44] S. Kim, S. I. Na, T. H. Kim, *et al.*, "Neural recording system with low-noise analog front-end and comparator-based cyclic ADC," in *SOC Conference (SOCC), 2012 IEEE International*, 2012, pp. 110-114.

[45] A. Moradi, M. Zgaren, and M. Sawan, "A 0.084 nJ/b FSK transmitter and 4.8 μW OOK receiver for ISM-band medical sensor networks," in *New Circuits and Systems Conference (NEWCAS), 2013 IEEE 11th International*, 2013, pp. 1-4.

[46] S. A. Mirbozorgi, H. Bahrami, M. Sawan, *et al.*, "A Smart Multicoil Inductively Coupled Array for Wireless Power Transmission," *Industrial Electronics, IEEE Transactions on*, vol. 61, pp. 6061-6070, 2014.

[47] A. Borna and K. Najafi, "A Low Power Light Weight Wireless Multichannel Microsystem for Reliable Neural Recording," *Solid-State Circuits, IEEE Journal of*, vol. 49, pp. 439-451, 2014.

[48] J. Tan, W. S. Liew, C. H. Heng, *et al.*, "A 2.4 GHz ULP Reconfigurable Asymmetric Transceiver for Single-Chip Wireless Neural Recording IC," *Biomedical Vircuits and Systems, IEEE Transactions on*, vol. 8, pp. 497-509, 2014.

[49] S. Gibson, J. W. Judy, and D. Markovic, "Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, pp. 469-478, 2010.

[50] C. Szi-Wen and C. Shih-Chieh, "Compressed sensing for Integral Pulse Frequency Modulation (IPFM)-based Heart Rate Variability spectral estimation," in *2012 34th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Piscataway, NJ, USA, 2012, pp. 5626-5629.

[51] Y. C. Liu, C. C. Lin, J.J. Tsai, *et al.*, "Model-Based Spike Detection of Epileptic EEG Data," *Sensors*, vol. 13, pp. 12536-12547, 2013.

[52] L. Orosco, A. G. Correa, and E. Laciar, "Review: a survey of performance and techniques for automatic epilepsy detection," *Journal of Medical and Biological Engineering*, vol. 33, pp. 526-537, 2013.

[53] L. Massi, M. Lagler, K. Hartwich, *et al.*, "Temporal dynamics of parvalbumin-expressing axo-axonic and basket cells in the rat medial prefrontal cortex in vivo," *The Journal of Neuroscience*, vol. 32, pp. 16496-16502, 2012.

[54] A. Rodriguez-Perez, J. Ruiz-Amaya, M. Delgado-Restituto, *et al.*, "A low-power programmable neural spike detection channel with embedded calibration and data compression," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 6, pp. 87-100, Apr. 2012.

[55] M. L. F. Janssen, D. G. M. Zwartjes, Y. Temel, *et al.*, "Subthalamic neuronal responses to cortical stimulation," *Movement Disorders*, vol. 27, pp. 435-438, 2012.

[56] A. Berényi, M. Belluscio, D. Mao, *et al.*, "Closed-loop control of epilepsy by transcranial electrical stimulation," *Science*, vol. 337, pp. 735-737, 2012.

[57] R. R. Harrison, R. J. Kier, C. A. Chestek, *et al.*, "Wireless Neural Recording With Single Low-Power Integrated Circuit," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, pp. 322-329, 2009.

[58] R. R. Harrison, P. T. Watkins, R. J. Kier, *et al.*, "A low-power integrated circuit for a wireless 100-electrode neural recording system," *Solid-State Circuits, IEEE Journal of*, vol. 42, pp. 123-33, Jan. 2007.

[59] M. S. Chae, Z. Yang, M. R. Yuce, *et al.*, "A 128-channel 6 mW wireless neural recording IC with spike feature extraction and UWB transmitter," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, pp. 312-321, 2009.

[60] A. M. Sodagar, G. E. Perlin, Y. Yao, *et al.*, "An implantable 64-channel wireless microsystem for single-unit neural recording," *Solid-State Circuits, IEEE Journal of*, vol. 44, pp. 2591-2604, 2009.

[61] U. Frey, U. Egert, F. Heer, *et al.*, "Microelectronic system for high-resolution mapping of extracellular electric fields applied to brain slices," *Biosensors and Bioelectronics*, vol. 24, pp. 2191-8, Mar. 2009.

[62] F. DePiero. "Digital Vs Analog Signal Processing." Internet: https://courseware.ee.calpoly.edu/~fdepiero/fdepiero_dsp_notes/dsp_notes.html [2015].

[63] X. L. Liu, H. J. Zhu, M. L. Zhang, *et al.*, "Design of a low-noise, high power efficiency neural recording front-end with an integrated real-time compressed sensing unit," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, 2015, pp. 2996-2999.

[64] J. Zhang, M. Srinjoy, Y. M. Suo, *et al.*, "A closed-loop compressive-sensing-based neural recording system," *Journal of Neural Engineering*, vol. 12 (036005), pp. 1-17, 2015.

[65] L. Duan, T. Wang, S. Wang, *et al.*, "A Wireless Neural Recording SoC and Implantable Microsystem Integration," Georgia Institute of Technology. Internet: https://smartech.gatech.edu/bitstream/handle/1853/54042/GT-CS-15-06.pdf [2015].

[66] T. K. T. Nguyen, Z. Navratilova, H. Cabral, *et al.*, "Closed-loop optical neural stimulation based on a 32-channel low-noise recording system with online spike sorting," *Journal of Neural Engineering*, vol. 11, p. 046005, 2014.

[67] P. Kmon, "Digitally assisted neural recording and spike detection multichannel integrated circuit designed in 180 nm CMOS technology," *Microelectronics Journal*, vol. 45, pp. 1187-1193, Sept. 2014.

[68] T. Morrison, M. Nagaraju, B. Winslow, *et al.*, "A 0.5 cm^3 Four-Channel 1.1 mW Wireless Biosignal Interface With 20 m Range," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 8, pp. 138-147, 2014.

[69] K. W. Cheng, X. D. Zou, J. H. Cheong, *et al.*, "100-Channel wireless neural recording system with 54-Mb/s data link and 40%-efficiency power link," in *Solid State Circuits Conference (A-SSCC), 2012 IEEE Asian*, 2012, pp. 185-188.

[70] K. Abdelhalim, H. M. Jafari, L. Kokarovtseva, *et al.*, "64-Channel UWB Wireless Neural Vector Analyzer SOC With a Closed-Loop Phase Synchrony-Triggered Neurostimulator," *Solid-State Circuits, IEEE Journal of*, vol. 48, pp. 2494-2510, 2013.

[71] Q. Chengliang, J. Shi, J. Parramon, *et al.*, "A Low-Power Configurable Neural Recording System for Epileptic Seizure Detection," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 7, pp. 499-512, 2013.

[72] M. Yin, D. A. Borton, J. Aceros, *et al.*, "A 100-channel hermetically sealed implantable device for wireless neurosensing applications," in *2012 IEEE International Symposium on Circuits and Systems, ISCAS 2012*, Seoul, Republic of Korea, 2012, pp. 2629-2632.

[73] L. Huang, X. Zhang, N. Guan, *et al.*, "Real-time multi-channel system for neural spikes acquisition and detection," in *2012 IEEE 10th International New Circuits and Systems Conference, NEWCAS 2012*, Montreal, QC, Canada, 2012, pp. 149-152.

[74] S. J. Thomas, R. R. Harrison, A. Leonardo, *et al.*, "A battery-free multichannel digital neural/EMG telemetry system for flying insects," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 6, pp. 424-436, 2012.

[75] R. R. Harrison, H. Fotowat, R. Chan, *et al.*, "Wireless Neural/EMG Telemetry Systems for Small Freely Moving Animals," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 5, pp. 103-111, 2011.

[76] R. F. Yazicioglu, K. Sunyoung, T. Torfs, *et al.*, "A 30 μW Analog Signal Processor ASIC for Portable Biopotential Signal Monitoring," *Solid-State Circuits, IEEE Journal of*, vol. 46, pp. 209-223, 2011.

[77] A. Bonfanti, G. Zambra, G. Baranauskas, *et al.*, "A wireless microsystem with digital data compression for neural spike recording," *Microelectronic Engineering*, vol. 88, pp. 1672-1675, 2011.

[78] A. Bonfanti, M. Ceravolo, G. Zambra, *et al.*, "A multi-channel low-power IC for neural spike recording with data compression and narrowband 400-MHz MC-FSK wireless transmission," in *ESSCIRC, 2010 Proceedings of the*, 2010, pp. 330-333.

[79] S. Farshchi, A. Pesterev, P. Nuyujukian, *et al.*, "Embedded Neural Recording With TinyOS-Based Wireless-Enabled Processor Modules," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, pp. 134-141, 2010.

[80] N. Verma, A. Shoeb, J. Bohorquez, *et al.*, "A Micro-Power EEG Acquisition SoC With Integrated Feature Extraction Processor for a Chronic Seizure Detection System," *Solid-State Circuits, IEEE Journal of*, vol. 45, pp. 804-816, 2010.

[81] J. N. Y. Aziz, K. Abdelhalim, R. Shulyzki, *et al.*, "256-Channel Neural Recording and Delta Compression Microsystem With 3D Electrodes," *Solid-State Circuits, IEEE Journal of*, vol. 44, pp. 995-1005, 2009.

[82] W. S. Liew, X. D. Zou, L. B. Yao, *et al.*, "A 1-V 60-μW 16-channel interface chip for implantable neural recording," in *Custom Integrated Circuits Conference, CICC '09. IEEE*, 2009, pp. 507-510.

[83] K. Donghwi, M. Stanacevic, R. Kamoua, *et al.*, "A low-power low-data-rate neural recording system with adaptive spike detection," in *Circuits and Systems, MWSCAS 2008. 51st Midwest Symposium on*, 2008, pp. 822-825.

[84] M. Mollazadeh, K. Murari, H. Schwerdt, *et al.*, "Wireless multichannel acquisition of neuropotentials," in *Biomedical Circuits and Systems Conference, BioCAS 2008. IEEE*, 2008, pp. 49-52.

[85] T. Borghi, A. Bonfanti, G. Zambra, *et al.*, "A Compact Multichannel System for Acquisition and Processing of Neural Signals," in *Engineering in Medicine and Biology*

Society, EMBS 2007. 29th Annual International Conference of the IEEE, 2007, pp. 441-444.

[86] J. Aziz, R. Karakiewicz, R. Genov, *et al.*, "In Vitro Epileptic Seizure Prediction Microsystem," in *Circuits and Systems, ISCAS 2007. IEEE International Symposium on*, 2007, pp. 3115-3118.

[87] W. JG, *Medical Instrumentation: Application And Design*. New York: John Wiley & Sons, Inc., 1998, pp. 121-176.

[88] S. Gibson, J. W. Judy, and D. Markovic, "Spike Sorting: The First Step in Decoding the Brain: The first step in decoding the brain," *Signal Processing Magazine, IEEE*, vol. 29, pp. 124-143, 2012.

[89] J. H. Byrne. "Resting Potentials and Action Potentials." Internet: http://neuroscience.uth.tmc.edu/s1/chapter01.html [2015, 10.19].

[90] H. Semmaoui, J. Drolet, A. Lakhssassi, *et al.*, "Setting Adaptive Spike Detection Threshold for Smoothed TEO Based on Robust Statistics Theory," *Biomedical Engineering, IEEE Transactions on*, vol. 59, pp. 474-482, 2012.

[91] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network: Computation in Neural Systems*, vol. 9, pp. 53-78, Nov. 1998.

[92] S. Gibson, J. W. Judy, and D. Markovic, "Comparison of spike-sorting algorithms for future hardware implementation," in *Engineering in Medicine and Biology Society, EMBS 2008. 30th Annual International Conference of the IEEE*, 2008, pp. 5015-5020.

[93] M. H. Zarifia, N. K. Ghalehjogh, and M. Baradaran-nia, "A new evolutionary approach for neural spike detection based on genetic algorithm," *Expert Systems with Applications*, vol. 42, pp. 462-467, Jan. 2015.

[94] X. Liu, X. Yang, and N. Zheng, "Automatic extracellular spike detection with piecewise optimal morphological filter," *Neurocomputing*, vol. 79, pp. 132-139, Mar. 2012.

[95] V. Karkare, S. Gibson, and D. Marković, "Energy-Efficient Digital Processing for Neural Action Potentials," in *Neural Computation, Neural Devices, and Neural Prosthesis*, Z. Yang, Ed. New York: Springer, 2014, pp. 23-40.

[96] H. L. Chan, M. A. Lin, T. Wu, *et al.*, "Detection of neuronal spikes using an adaptive threshold based on the max–min spread sorting method," *Journal of Neuroscience Methods*, vol. 172, pp. 112-121, July 2008.

[97] K. S. Guillory and R. A. Normann, "A 100-channel system for real time detection and storage of extracellular spike waveforms," *Journal of Neuroscience Methods*, vol. 91, pp. 21-29, Sept.1999.

[98] D. L. Donoho, "De-noising by soft-thresholding," *Information Theory, IEEE Transactions on*, vol. 41, pp. 613-627, 1995.

[99] Z. Yang, W. Liu, M. R. Keshtkaran, *et al.*, "A new EC–PC threshold estimation method for in vivo neural spike detection," *Journal of neural engineering*, vol. 9(046017), pp. 1-16, 2012.

[100] P. H. Thakur, H. Lu, S. S. Hsiao, *et al.*, "Automated optimal detection and classification of neural action potentials in extra-cellular recordings," *Journal of Neuroscience Methods*, vol. 162, pp. 364-376, May 2007.

[101] A. Guanglei, D. Kanishka, H. Cheng, *et al.*, "An analog front-end circuit with spike detection for implantable neural recording system design," in *Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on*, 2014, pp. 881-884.

[102] N. Li, H. Semmaoui, and M. Sawan, "Modified Maximum and Minimum Spread estimation method for detection of neural spikes," in *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*, 2013, pp. 530-533.

[103] S. Mukhopadhyay and G. Ray, "A new interpretation of nonlinear energy operator and its efficacy in spike detection," *Biomedical Engineering, IEEE Transactions on*, vol. 45, pp. 180-187, 1998.

[104] J. Drolet, H. Semmaoui, and M. Sawan, "Low-power energy-based CMOS digital detector for neural recording arrays," in *Biomedical Circuits and Systems Conference (BioCAS), 2011 IEEE*, 2011, pp. 13-16.

[105] V. Karkare, S. Gibson, and D. Markovic, "A 130-W, 64-channel neural spike-sorting DSP chip," *Solid-State Circuits, IEEE Journal of*, vol. 46, pp. 1214-1222, 2011.

[106] B. Gosselin and M. Sawan, "An ultra low-power CMOS automatic action potential detector," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, pp. 346-353, 2009.

[107] I. N. Bankman, K. O. Johnson, and W. Schneider, "Optimal detection, classification, and superposition resolution in neural waveform recordings," *Biomedical Engineering, IEEE Transactions on*, vol. 40, pp. 836-841, 1993.

[108] H. Kaneko, S. S. Suzuki, J. Okada, *et al.*, "Multineuronal spike classification based on multisite electrode recording, whole-waveform analysis, and hierarchical clustering," *Biomedical Engineering, IEEE Transactions on*, vol. 46, pp. 280-290, 1999.

[109] I. Obeid and P. D. Wolf, "Evaluation of spike-detection algorithms fora brain-machine interface application," *Biomedical Engineering, IEEE Transactions on*, vol. 51, pp. 905-911, 2004.

[110] T. Sato, T. Suzuki, and K. Mabuchi, "Fast Template Matching for Spike Sorting," *Electronics, Information and Systems, IEEJ Transactions on*, vol. 127, pp. 1680-1685, 2007.

[111] K. Oweiss and M. Aghagolzadeh, "Detection and classification of extracellular action potential recordings," in *Statistical Signal Processing for Neuroscience and Neurotechnology*, Amsterdam: Elsevier, 2010, pp. 15-74.

[112] W. J. Hwang, S. H. Wang, and Y. T. Hsu, "Spike Detection Based on Normalized Correlation with Automatic Template Generation," *Sensors*, vol. 14, pp. 11049-11069, June 2014.

[113] T. Haga, O. Fukayama, Y. Takayama, *et al.*, "Efficient sequential Bayesian inference method for real-time detection and sorting of overlapped neural spikes," *Journal of Neuroscience Methods*, vol. 219, pp. 92-103, Sept. 2013.

[114] F. Franke, R. Quian Quiroga, A. Hierlemann, *et al.*, "Bayes optimal template matching for spike sorting – combining fisher discriminant analysis with optimal filtering," *Journal of Computational Neuroscience*, vol. 38, pp. 439-459, June 2015.

[115] Y. Yuan, C. Yang, and J. Si, "The M-Sorter: An automatic and robust spike detection and classification system," *Journal of Neuroscience Methods*, vol. 210, pp. 281-290, Sept. 2012.

[116] U. Rutishauser, E. M. Schuman, and A. N. Mamelak, "Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo," *Journal of Neuroscience Methods*, vol. 154, pp. 204-224, June 2006.

[117] S. Kim and J. McNames, "Automatic spike detection based on adaptive template matching for extracellular neural recordings," *Journal of Neuroscience Methods*, vol. 165, pp. 165-174, Sept. 2007.

[118] S. M. Thurman and H. Lu, "Bayesian integration of position and orientation cues in perception of biological and non-biological forms," *Frontiers in Human Neuroscience*, vol. 8, p. 91, Feb. 2014.

[119] M. A. Nicolelis, "Actions from thoughts," *Nature*, vol. 409, pp. 403-407, 2001.

[120] T. W. Berger, A. Ahuja, S. H. Courellis, *et al.*, "Restoring lost cognitive function," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 24, pp. 30-44, 2005.

[121] A. Pavlov, V. A. Makarov, I. Makarova, *et al.*, "Sorting of neural spikes: when wavelet based methods outperform principal component analysis," *Natural Computing*, vol. 6, pp. 269-281, 2007.

[122] S. E. Paraskevopoulou, D. Y. Barsakcioglu, M. R. Saberi, *et al.*, "Feature extraction using first and second derivative extrema (FSDE) for real-time and hardware-efficient spike sorting," *Journal of Neuroscience Methods*, vol. 215, pp. 29-37, Apr. 2013.

[123] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural computation*, vol. 16, pp. 1661-1687, Aug. 2004.

[124] H. Abdi and P. Molin, "Lilliefors/Van Soest's test of normality," *Encyclopedia of measurement and statistics*, pp. 540-544, 2007.

[125] C. Zhang, X. Zhang, M. Q. Zhang, *et al.*, "Neighbor number, valley seeking and clustering," *Pattern Recognition Letters*, vol. 28, pp. 173-180, Jan. 2007.

[126] A. M. Kamboh and A. J. Mason, "Computationally Efficient Neural Feature Extraction for Spike Sorting in Implantable High-Density Recording Systems," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 21, pp. 1-9, 2013.

[127] Y. Q. Li, Z. L. Yu, N. Bi, *et al.*, "Sparse Representation for Brain Signal Processing: A tutorial on methods and applications," *Signal Processing Magazine, IEEE*, vol. 31, pp. 96-106, 2014.

[128] M. Shoaib, N. K. Jha, and N. Verma, "Signal Processing With Direct Computations on Compressively Sensed Data," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 23, pp. 30-43, 2015.

[129] B. D. He, A. Wein, L. R. Varshney, *et al.*, "Generalized Analog Thresholding for Spike Acquisition at Ultra-Low Sampling Rates," *Journal of Neurophysiology*, vol.114, pp. 746-760, 2015.

[130] Y. C. Eldar and G. Kutyniok, *Compressed sensing: theory and applications*. Cambridge UK: Cambridge University Press, 2012, pp. 1- 303.

[131] G. Peyre, "Best basis compressed sensing," *Signal Processing, IEEE Transactions on*, vol. 58, pp. 2613-2622, 2010.

[132] C. Caratheodory, "Uber den Variabilitatsbereich der Koeffizienten von Potenzreihen,die gegebene Werte nicht annehmen," *Mathematische Annalen*, vol. 64, pp. 95-115, 1907.

[133] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *Information Theory, IEEE Transactions on*, vol. 52, pp. 489-509, 2006.

[134] F. Chen, A. P. Chandrakasan, and V. M. Stojanovic, "Design and Analysis of a Hardware-Efficient Compressed Sensing Architecture for Data Compression in Wireless Sensors," *Solid-State Circuits, IEEE Journal of*, vol. 47, pp. 744-756, 2012.

[135] A. G. Dimakis, R. Smarandache, and P. O. Vontobel, "LDPC Codes for Compressed Sensing," *Information Theory, IEEE Transactions on*, vol. 58, pp. 3093-3114, 2012.

[136] E. J. Candes and T. Tao, "Decoding by linear programming," *Information Theory, IEEE Transactions on*, vol. 51, pp. 4203-4215, 2005.

[137] M. Rudelson and R. Vershynin, "Sparse reconstruction by convex relaxation: Fourier and Gaussian measurements," in *Information Sciences and Systems, 2006 40th Annual Conference on*, 2006, pp. 207-212.

[138] M. Shoaran, M. H. Kamal, C. Pollo, *et al.*, "Compact Low-Power Cortical Recording Architecture for Compressive Multichannel Data Acquisition," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 8, pp. 857-870, 2014.

[139] Z. L. Zhang, T. P. Jung, S. Makeig, *et al.*, "Compressed Sensing of EEG for Wireless Telemonitoring With Low Energy Consumption and Inexpensive Hardware," *Biomedical Engineering, IEEE Transactions on*, vol. 60, pp. 221-224, 2013.

[140] M. F. Duarte and Y. C. Eldar, "Structured Compressed Sensing: From Theory to Applications," *Signal Processing, IEEE Transactions on*, vol. 59, pp. 4053-4085, 2011.

[141] S. X. Li, F. Gao, G. N. Ge, *et al.*, "Deterministic Construction of Compressed Sensing Matrices via Algebraic Curves," *Information Theory, IEEE Transactions on*, vol. 58, pp. 5035-5041, 2012.

[142] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, vol. 2, pp. 183-202, 2009.

[143] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, pp. 3397-3415, 1993.

[144] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, pp. 265-274, Nov. 2009.

[145] Z. Charbiwala, P. Martin, and M. B. Srivastava, "CapMux: A scalable analog front end for low power compressed sensing," in *Green Computing Conference (IGCC), 2012 International*, 2012, pp. 1-10.

[146] Z. Charbiwala, V. Karkare, S. Gibson, *et al.*, "Compressive Sensing of Neural Action Potentials Using a Learned Union of Supports," in *Body Sensor Networks (BSN), 2011 International Conference on*, 2011, pp. 53-58.

[147] M. Shaou-Gang and C. Shu-Nien, "Wavelet-based lossy-to-lossless ECG compression in a unified vector quantization framework," *Biomedical Engineering, IEEE Transactions on*, vol. 52, pp. 539-543, 2005.

[148] H. Mamaghanian, N. Khaled, D. Atienza, *et al.*, "Design and Exploration of Low-Power Analog to Information Conversion Based on Compressed Sensing," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 2, pp. 493-501, 2012.

[149] J. A. Tropp, J. N. Laska, M. F. Duarte, *et al.*, "Beyond Nyquist: Efficient Sampling of Sparse Bandlimited Signals," *Information Theory, IEEE Transactions on*, vol. 56, pp. 520-544, Jan. 2010.

[150] J. A. Tropp, M. B. Wakin, M. F. Duarte, *et al.*, "Random filters for compressive sampling and reconstruction," *2006 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol III , pp. 872 - 875, 2006.

[151] M. Mishali and Y. C. Eldar, "From Theory to Practice: Sub-Nyquist Sampling of Sparse Wideband Analog Signals," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, pp. 375-391, 2010.

[152] J. Romberg, "Compressive Sensing by Random Convolution," *Siam Journal on Imaging Sciences*, vol. 2, pp. 1098-1128, 2009.

[153] J. P. Slavinsky, J. N. Laska, M. A. Davenport, *et al.*, "The compressive multiplexer for multi-channel compressive sensing," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2011, pp. 3980-3983.

[154] X. Chen, Z. Z. Yu, S. Hoyos, *et al.*, "A Sub-Nyquist Rate Sampling Receiver Exploiting Compressive Sensing," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, pp. 507-520, 2011.

[155] W. Z. Lu, K. Kpalma, and J. Ronsin, "Sparse Binary Matrices of LDPC Codes for Compressed Sensing," in *2012 Data Compression Conference (DCC)*, 2012, pp. 405-405.

[156] Y. M. Suo, J. Zhang, T. Xiong, *et al.*, "Energy-Efficient Multi-Mode Compressed Sensing System for Implantable Neural Recordings," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 8, pp. 648-659, 2014.

[157] T. Xiong, Y. M. Suo, J. Zhang, *et al.*, "A dictionary learning algorithm for multi-channel neural recordings," in *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*, 2014, pp. 9-12.

[158] J. Zhang, Y. M. Suo, S. Mitra, *et al.*, "Reconstruction of neural action potentials using signal dependent sparse representations," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, 2013, pp. 1520-1523.

[159] D. Bellasi, R. Rovatti, L. Benini, *et al.*, "An architecture for low-power compressed sensing and estimation in wireless sensor nodes," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, 2014, pp. 1732-1735.

[160] V. Karkare, S. Gibson, and D. Markovic, "A 130-µW, 64-channel spike-sorting DSP chip," in *Solid-State Circuits Conference, A-SSCC 2009. IEEE Asian*, 2009, pp. 289-292.

[161] A. C. Gilbert, M. J. Strauss, J. A. Tropp, *et al.*, "One sketch for all: fast algorithms for compressed sensing," presented at the Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, San Diego, California, USA, 2007, pp. 237-246.

[162] X. J. Liu and S. T. Xia, "Reconstruction guarantee analysis of binary measurement matrices based on girth," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, 2013, pp. 474-478.

[163] L. Applebaum, S. D. Howard, S. Searle, *et al.*, "Chirp sensing codes: Deterministic compressed sensing measurements for fast recovery," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 283-290, Mar. 2009.

[164] A. Amini and F. Marvasti, "Deterministic Construction of Binary, Bipolar, and Ternary Compressed Sensing Matrices," *Information Theory, IEEE Transactions on*, vol. 57, pp. 2360-2370, 2011.

[165] N. Yu and Y. Li, "Deterministic construction of Fourier-based compressed sensing matrices using an almost difference set," *EURASIP Journal on Advances in Signal Processing*, vol. 2013, pp. 1-14, Oct. 2013.

[166] A. M. Kamboh, K. G. Oweiss, and A. J. Mason, "Resource constrained VLSI architecture for implantable neural data compression systems," in *Circuits and Systems, ISCAS 2009. IEEE International Symposium on*, 2009, pp. 1481-1484.

[167] R. H. Olsson and K. D. Wise, "A three-dimensional neural recording microsystem with implantable data compression circuitry," *Solid-State Circuits, IEEE Journal of*, vol. 40, pp. 2796-2804, 2005.

[168] V. Karkare, S. Gibson, and D. Markovic, "A 75-µW, 16-Channel Neural Spike-Sorting Processor With Unsupervised Clustering," *Solid-State Circuits, IEEE Journal of*, vol. 48, pp. 2230-2238, 2013.

[169] V. Shalchyan, W. Jensen, and D. Farina, "Spike Detection and Clustering With Unsupervised Wavelet Optimization in Extracellular Neural Recordings," *Biomedical Engineering, IEEE Transactions on*, vol. 59, pp. 2576-2585, 2012.

[170] F. Franke, M. Natora, C. Boucsein, *et al.*, "An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes," *Journal of Computational Neuroscience*, vol. 29, pp. 127-148, Aug. 2010.

[171] A. Meraoumia, S. Chitroub, and A. Bouridane, "2D and 3D palmprint information, PCA and HMM for an improved person recognition performance," *Integrated Computer-Aided Engineering*, vol. 20, pp. 303-319, 2013.

[172] E. Gokgoz and A. Subasi, "Comparison of decision tree algorithms for EMG signal classification using DWT," *Biomedical Signal Processing and Control*, vol. 18, pp. 138-144, Apr. 2015.

[173] S. C. Wu, A. L. Swindlehurst, and Z. Nenadic, "A novel framework for feature extraction in multi-sensor action potential sorting," *Journal of Neuroscience Methods*, vol. 253, pp. 262-271, Sept. 2015.

[174] M. Zamani and A. Demosthenous, "Feature Extraction Using Extrema Sampling of Discrete Derivatives for Spike Sorting in Implantable Upper-Limb Neural Prostheses," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 22, pp. 716-726, 2014.

[175] E. J. Zuperku, I. Prkic, A. G. Stucke, *et al.*, "Automatic classification of canine PRG neuronal discharge patterns using K-means clustering," *Respiratory Physiology & Neurobiology*, vol. 207, pp. 28-39, Feb. 2015.

[176] J. T. Moyer, S. F. Danish, J. G. Keating, *et al.*, "Implementation of dual simultaneous microelectrode recording systems during deep brain stimulation surgery for Parkinson's disease: technical note," *Operative Neurosurgery*, vol. 60, pp. E177 - E178, Feb. 2007. DOI:10.1227/01.NEU.0000249250.40676.7E.

[177] J. Neimat, C. Hamani, P. Giacobbe, *et al.*, "Neural stimulation successfully treats depression in patients with prior ablative cingulotomy," *Am J Psychiatry*, vol. 165, pp. 687-93, 2008.

[178] F. Shahrokhi, K. Abdelhalim, D. Serletis, *et al.*, "The 128-Channel Fully Differential Digital Integrated Neural Recording and Stimulation Interface," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 4, pp. 149-161, 2010.

[179] G. Charvet, L. Rousseau, O. Billoint, *et al.*, "BioMEA: a versatile high-density 3D microelectrode array system using integrated electronics," *Biosensors and Bioelectronics*, vol. 25, pp. 1889-96, Apr. 2010.

[180] E. J. Candes and Y. Plan, "Near-ideal model selection by ℓ_1 minimization," *The Annals of Statistics*, vol. 37, pp. 2145-2177, Oct. 2009.

[181] X. Y. Zhang, J. T. Wen, Y. X. Han, *et al.*, "An improved compressive sensing reconstruction algorithm using linear/non-linear mapping," in *2011 Information Theory and Applications Workshop (ITA)*, 2011, pp. 1-7.

[182] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 301-321, May 2009.

[183] T. Blumensath and M. Davies, "Iterative Thresholding for Sparse Approximations," *Journal of Fourier Analysis and Applications*, vol. 14, pp. 629-654, Dec. 2008.

[184] Z. Zhang and B. D. Rao, "Extension of SBL Algorithms for the Recovery of Block Sparse Signals With Intra-Block Correlation," *Signal Processing, IEEE Transactions on*, vol. 61, pp. 2009-2015, 2013.

[185] B. Liu, Z. Zhang, G. Xu, *et al.*, "Energy efficient telemonitoring of physiological signals via compressed sensing: A fast algorithm and power consumption evaluation," *Biomedical Signal Processing and Control*, vol. 11, pp. 80-88, May 2014.

[186] R. G. Baraniuk, V. Cevher, M. F. Duarte, *et al.*, "Model-Based Compressive Sensing," *Information Theory, IEEE Transactions on*, vol. 56, pp. 1982-2001, 2010.

[187] S. D. Howard, A. R. Calderbank, and S. J. Searle, "A fast reconstruction algorithm for deterministic compressive sensing using second order reed-muller codes," in *Information Sciences and Systems, CISS 2008. 42nd Annual Conference on*, 2008, pp. 11-15.

[188] R. Baraniuk, M. Davenport, R. DeVore, *et al.*, "A Simple Proof of the Restricted Isometry Property for Random Matrices," *Constructive Approximation*, vol. 28, pp. 253-263, 2008/12/01 2008.

[189] R. Calderbank, S. Howard, and S. Jafarpour, "Construction of a Large Class of Deterministic Sensing Matrices That Satisfy a Statistical Isometry Property," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, pp. 358-374, 2010.

[190] J. Martinez, C. Pedreira, M. J. Ison, *et al.* "Dataset # 3: Simulated extracellular recordings."Internet:www2.le.ac.ul/departments/engineering/research/bioengineering/neuroengineering-lab/software [2015]

[191] Z. L. Zhang, T. P. Jung, S. Makeig, *et al.*, "Compressed Sensing for Energy-Efficient Wireless Telemonitoring of Noninvasive Fetal ECG Via Block Sparse Bayesian Learning," *Biomedical Engineering, IEEE Transactions on*, vol. 60, pp. 300-309, 2013.

[192] D. Donoho, V. Stodden, and Y. Tsaig. (2007) "Sparse Lab." Internet: http://sparselab.stanford.edu/ [2015].

[193] E. v. den Berg and M. P. Friedlander. (2013). "SPGL1: A solver for large-scale sparse reconstruction." Internet: https://www.math.ucdavis.edu/~mpf/spgl1/ [2015].

[194] N. C. Klapoetke, Y. Murata, S. S. Kim, *et al.*, "Independent optical excitation of distinct neural populations," *Nature Methods*, vol. 11, pp. 338-346, Mar. 2014.

[195] D. Khodagholy, J. N. Gelinas, T. Thesen, *et al.*, "NeuroGrid: recording action potentials from the surface of the brain," *Nature Neuroscience*, vol. 18, pp. 310-315, Feb. 2015.

[196] M. A. J. Lourens, H. G. E. Meijer, M. F. Contarino, *et al.*, "Functional neuronal activity and connectivity within the subthalamic nucleus in Parkinson's disease," *Clinical Neurophysiology*, vol. 124, pp. 967-981, May 2013.

[197] Y. C. Yang, C. H. Tai, M. K. Pan, *et al.*, "The T- type calcium channel as a new therapeutic target for Parkinson's disease," *Pflügers Archiv - European Journal of Physiology*, vol. 466, pp. 747-755, Apr. 2014.

[198] D. Han, Y. J. Zheng, R. RajKumar, *et al.*, "A 0.45 V 100-Channel Neural-Recording IC With Sub-μW/Channel Consumption in 0.18 μm CMOS," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 7, pp. 735-746, 2013.

[199] T. A. Szuts, V. Fadeyev, S. Kachiguine, *et al.*, "A wireless multi-channel neural amplifier for freely moving animals," *Nature Neuroscience*, vol. 14, pp. 263-269, Feb. 2011.

[200] Z. Yang, Q. Zhao, E. Keefer, *et al.*, "Noise characterization, modeling, and reduction for in vivo neural recording," in *Advances in neural information processing systems*, 2009, pp. 2160-2168.

[201] E. N. Brown, R. E. Kass, and P. P. Mitra, "Multiple neural spike train data analysis: state-of-the-art and future challenges," *Nature Neuroscience*, vol. 7, pp. 456-461, May 2004.

[202] J. Wild, Z. Prekopcsak, T. Sieger, *et al.*, "Performance comparison of extracellular spike sorting algorithms for single-channel recordings," *Journal of Neuroscience Methods*, vol. 203, pp. 369-376, Jan. 2012.

[203] N. Thanh, A. Khosravi, I. Hettiarachchi, *et al.*, "Classification of neural action potentials using mean shift clustering," in *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, 2014, pp. 1247-1252.

[204] S. E. Paraskevopoulou, D. Wu, A. Eftekhar, *et al.*, "Hierarchical Adaptive Means (HAM) clustering for hardware-efficient, unsupervised and real-time spike sorting," *Journal of Neuroscience Methods*, vol. 235, pp. 145-156, Sept. 2014.

[205] P. T. Watkins, G. Santhanam, K. V. Shenoy, *et al.*, "Validation of adaptive threshold spike detector for neural recording," in *Engineering in Medicine and Biology Society, IEMBS '04. 26th Annual International Conference of the IEEE*, 2004, pp. 4079-4082.

[206] U. Iruansi and S. Apeh, "Optimal Spike Detection Technique Based on Amplitude Threshold," *British Journal of Applied Science & Technology*, vol. 4, pp. 3539 - 3549, 2014.

[207] R. J. Brychta, R. Shiavi, D. Robertson, *et al.*, "Spike detection in human muscle sympathetic nerve activity using the kurtosis of stationary wavelet transform coefficients," *Journal of Neuroscience Methods*, vol. 160, pp. 359-367, Mar. 2007.

[208] S. Kim, J. McNames, and K. Burchiel, "Action potential detection with automatic template matching," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Proceedings*, 2004, pp. 41-44.

[209] C. Ekanadham, D. Tranchina, and E. P. Simoncelli, "A unified framework and method for automatic neural spike identification," *Journal of Neuroscience Methods*, vol. 222, pp. 47-55, Jan. 2014.

[210] V. Shalchyan and D. Farina, "A non-parametric Bayesian approach for clustering and tracking non-stationarities of neural spikes," *Journal of Neuroscience Methods*, vol. 223, pp. 85-91, Feb. 2014.

[211] Y. Zhou, T. Wu, A. Rastegarnia, *et al.*, "On the robustness of EC–PC spike detection method for online neural recording," *Journal of Neuroscience Methods*, vol. 235, pp. 316-330, Sept. 2014.

[212] Z. Q. Dong, H. Y. Gu, Y. Wan, *et al.*, "Wireless body area sensor network for posture and gait monitoring of individuals with Parkinson's disease," in *Networking, Sensing and Control (ICNSC), 2015 IEEE 12th International Conference on*, 2015, pp. 81 - 86.

[213] G. Valenza, M. Nardelli, A. Lanata, *et al.*, "Wearable Monitoring for Mood Recognition in Bipolar Disorder Based on History-Dependent Long-Term Heart Rate Variability Analysis," *Biomedical and Health Informatics, IEEE Journal of*, vol. 18, pp. 1625-1635, 2014.

[214] J. Jeppesen, S. Beniczky, P. Johansen, *et al.*, "Exploring the capability of wireless near infrared spectroscopy as a portable seizure detection device for epilepsy patients," *Seizure*, vol. 26, pp. 43-48, Mar. 2015.

[215] D. Han, Y. Zheng, R. Rajkumar, *et al.*, "A 0.45 V 100-channel neural-recording IC with sub-μW/channel consumption in 0.18μm CMOS," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, 2013, pp. 290-291.

[216] A. Berényi, Z. Somogyvári, A. J. Nagy, *et al.*, "Large-scale, high-density (up to 512 channels) recording of local circuits in behaving animals," *Journal of neurophysiology*, vol. 111, pp. 1132-1149, 2014.

[217] K. Hyejung, K. Sunyoung, N. Van Helleputte, *et al.*, "A Configurable and Low-Power Mixed Signal SoC for Portable ECG Monitoring Applications," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 8, pp. 257-267, 2014.

[218] B. Gosselin, "Recent Advances in Neural Recording Microsystems," *Sensors*, vol. 11, pp. 4572-4597, 2011.

[219] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM journal on scientific computing*, vol. 20, pp. 33-61, 1998.

[220] M. Lin, R. Jin, and C. S. Zhang, "Efficient sparse recovery via adaptive non-convex regularizers with oracle property," in *Uncertainty in Artificial Intelligence - Proceedings of the 30th Conference, UAI 2014*, Quebec City, Quebec, Canada, 2014.

[221] A. Shirazinia and S. Dey, "Optimized compressed sensing matrix design for noisy communication channels," in *Communications (ICC), 2015 IEEE International Conference on*, 2015, pp. 4547-4552.

[222] R. Saab and Ö. Yilmaz, "A short note on non-convex compressed sensing," in *SAMPTA'09, 8th international conference on Sampling Theory and Applications*, Marseille, France, 2009.

[223] A. M. Dixon, E. G. Allstot, D. Gangopadhyay, *et al.*, "Compressed sensing system considerations for ECG and EMG wireless biosensors," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 6, pp. 156-166, 2012.

[224] N. Li and M. Sawan, "Neural signal compression using a minimum Euclidean or Manhattan distance cluster-based deterministic compressed sensing matrix," *Biomedical Signal Processing and Control*, vol. 19, pp. 44-55, May 2015.

[225] A. Bonfanti, M. Ceravolo, G. Zambra, *et al.*, "A multi-channel low-power system-on-chip for single-unit recording and narrowband wireless transmission of neural signal," in *2010 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, Buenos Aires, Argentina, 2010, pp. 1555-1560.

[226] R. R. Harrison, "A low-power integrated circuit for adaptive detection of action potentials in noisy signals," in *Engineering in Medicine and Biology Society, Proceedings of the 25th Annual International Conference of the IEEE*, 2003, pp. 3325-3328.

[227] B. Gosselin, A. E. Ayoub, J. F. Roy, *et al.*, "A mixed-signal multichip neural recording interface with bandwidth reduction," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 3, pp. 129-141, 2009.

APPENDIX A – COMPLEMENTARY BACKGROUND ON COMPRESSED SENSING THEORY

A.1. Vector Space

In the case of a discrete, finite domain, signals can be viewed as vectors in an *n*-dimensional Euclidean space, and the ℓ_p norms are frequently used for the measure. ℓ_p norms can be defined as (A.1),

$$\|x\|_p = \begin{cases} (\sum_{i=1}^{n} |x_i|^p)^{1/p}, & p \in (0,\infty) \\ \max\limits_{i=1,2,\ldots,n} |x_i|, & p = \infty \end{cases} \tag{A.1}$$

A set $\{v_1, v_2 \cdots, v_n\}$ is called a basis for $\mathbb{R}^n$ if the vectors in the set span $\mathbb{R}^n$ and are linearly independent. Each vector in the space has a unique representation as a linear combination of these basis vectors. Regarding an $x \in \mathbb{R}^n$, there exist coefficients $a = \{a_1, a_2, \cdots, a_n\}$ to form (A.2),

$$x = \sum_{i=1}^{n} a_i v_i \tag{A.2}$$

Note that $\{v_1, v_2 \cdots, v_n\}$ comprises an $n \times n$ matrix *V*, so (A.2) can be written as (A.3),

$$x = Va \tag{A.3}$$

For a basis $\{v_1, v_2 \cdots, v_n\}$ and every entry of this vector, if (A.4) is satisfied,

$$\langle v_i, v_j \rangle = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{A.4}$$

then $\{v_1, v_2 \cdots, v_n\}$ is called an orthonormal basis.

If a set vectors $\{v_1', v_2' \cdots, v_n'\}$ in $\mathbb{R}^m$, where $m < n$, comprise a matrix V', such that for all vector $x \in \mathbb{R}^m$,

$$C\|x\|_2^2 \leq \|V'^T x\|_2^2 \leq D\|x\|_2^2 \tag{A.5}$$

where $C, D \in (0,\infty)$, then $\{v_1', v_2' \cdots, v_n'\}$ calls a frame. If *C* and *D* can be chosen as $C = D$, then the frame is called *A*-tight. If *C* = *D* = 1, then *V'* is a parseval frame.

A.2. Sensing Matrix

(*Johnson-Lindenstruss Lemma*) Let $\varepsilon \in (0,1)$ be given. For every set P of $N(P)$ points in $\mathbb{R}^N$, if n is a positive integer such that $n > n_0 = O(\ln(N(P))/\varepsilon^2)$, there exists a Lipschitz mapping $f: \mathbb{R}^N \to \mathbb{R}^n$ such that (A.6),

$$(1-\varepsilon)\|u-v\|_{l_p}^2 \leq \|f(u)-f(v)\|_{l_p}^2 \leq (1+\varepsilon)\|u-v\|_{l_p}^2 \tag{A.6}$$

For all $u, v \in P$.

Let Φ be a random matrix of size $n \times N$ drawn according to any distribution that satisfies the concentration inequality. Then, for any set T with $N(T) = k < n$ and any $0 < \varepsilon < 1$, we have (A.7),

$$(1-\varepsilon)\|x\|_{l_p} \leq \|\Phi x\|_{l_p} \leq (1+\varepsilon)\|x\|_{l_p}, \text{ for all } x \in X_T \tag{A.7}$$

with probability

$$\geq 1 - 2(12/\varepsilon)^k \exp(-c_0(\varepsilon/2)n) \tag{A.8}$$

The concentration equality is defined as (A.9),

$$\Pr(|\|\Phi x\|_{l_p}^2 - \|x\|_{l_p}^2| \geq \varepsilon\|x\|_{l_p}^2) \leq 2\exp(-nc_0(\varepsilon)) \tag{A.9}$$

A matrix Φ satisfies the null space property (NSP) of order k if there exists a constant $C>0$ such that,

$$\|y_\Lambda\|_2 \leq C\frac{\|y_{\Lambda^c}\|_1}{\sqrt{k}} \tag{A.10}$$

Holds for all $y \in \text{Null}(\Phi)$ and for all Λ such that $|\Lambda| \leq k$, $\text{Null}(\Phi) = \{z: \Phi z = 0\}$. $\Lambda \subset \{1, 2, \cdots, n\}$ is a subset of indices and $\Lambda^c \subset \{1, 2, \cdots, n\} \backslash \Lambda$. If a vector y is exactly k-sparse, then there exists a Λ such that $\|y_{\Lambda^c}\|_1 = 0$ and implies that $y_\Lambda = 0$.

Defining $F_1: \mathbb{R}^m \to \mathbb{R}^n$. (A.10) can be changed to be (A.11),

$$C\|F_1(\Phi x)_1 - x\|_2 \leq C\frac{\sigma_k(x)_1}{\sqrt{k}} \tag{A.11}$$

For all x, where $\sigma_k(x)_1$ is defined in (2.16). This guarantees exact recovery of all possible k-sparse signals.

A.3. Signal Recovery In Noise

For the signal recovery, ℓ_1 minimization is applied, which is expressed as (A.12)

$$x' = \arg\min_z \|z\|_1, \text{ subject to } z \in f(y) = \{z: Az = y\}, \tag{A.12}$$

The bounded and Gaussian noisy signal recoveries are listed as follows.

Suppose that Φ satisfies the RIP of order $2k$ with $\varepsilon_{2k} < \sqrt{2} - 1$, and let $y = Ax + B$ where $\|B\|_2 \leq \gamma$, the solution *x'* to (A.12) obeys (A.13),

$$\|x' - x\|_2 \leq C_0 \frac{\sigma_k(x)_1}{\sqrt{k}} + C_1 \gamma \tag{A.13}$$

where $C_0 = 2\frac{1-(1-\sqrt{2})\varepsilon_{2k}}{1-(1+\sqrt{2})\varepsilon_{2k}}$ and $C_1 = 4\frac{\sqrt{1+\varepsilon_{2k}})}{1-(1+\sqrt{2})\varepsilon_{2k}}$.

Suppose that Φ satisfies the RIP of order $2k$ with $\varepsilon_{2k} < \sqrt{2} - 1$. Moreover, suppose $x \in \sum_k$ and that the measurement can be expressed as $y = Ax + B$ where B obeys the Gaussian distribution, that is, $B \sim N(0, \sigma^2)$. When $f(y) = \{z: \|Az - y\|_2 \leq 2\sqrt{m}\sigma\}$, the solution *x'* to (A.12) obeys (A.14),

$$\|x' - x\|_2 \leq 8\frac{\sqrt{1+\varepsilon_{2k}})}{1-(1+\sqrt{2})\varepsilon_{2k}}\sqrt{m}\sigma \tag{A.14}$$

With probability at least $1 - \exp(-c_0 m)$.

APPENDIX B – IMPLEMENTATION OF THE FRONT-END CIRCUIT

The built front-end circuit including signal filtering, amplifier and ADC is shown in B.1.

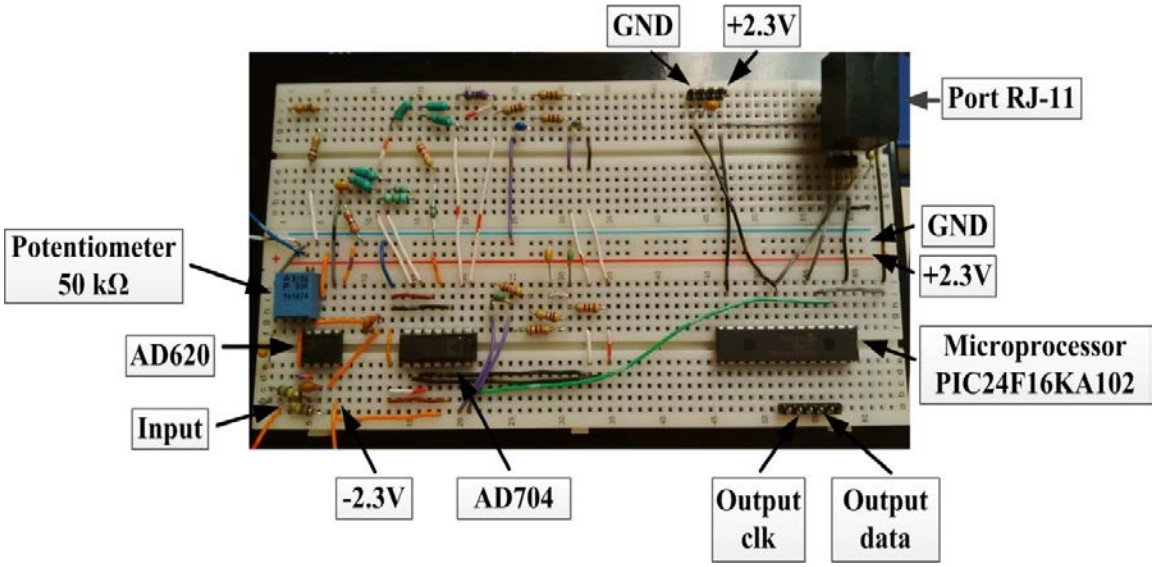


Figure B.1 The front-end circuit for the proposed signal processing system

The result of signal amplification for a signal of 100 mV with the gain of 10 is shown in B.2.

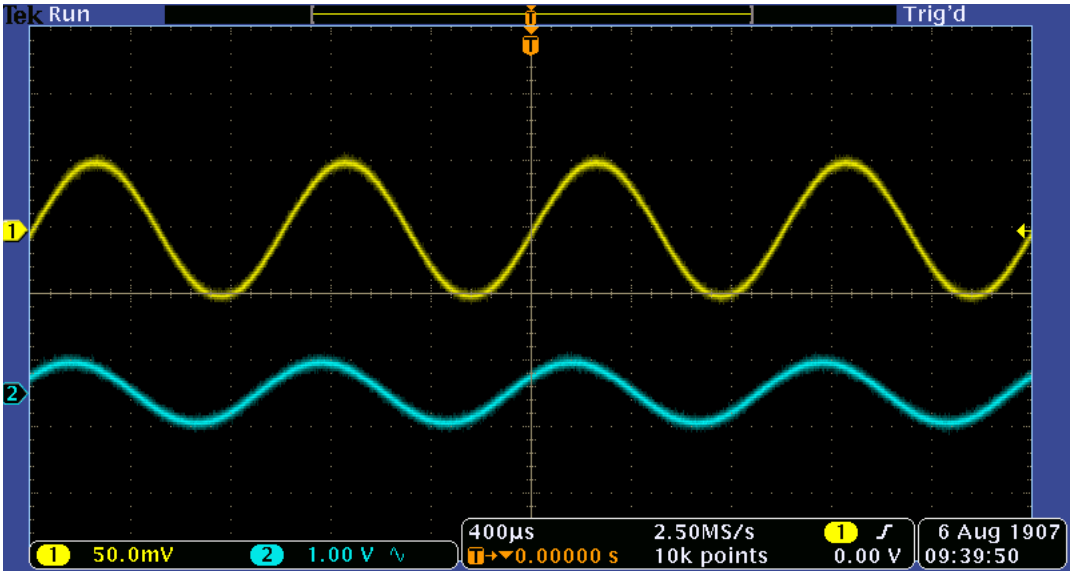


Figure B.2 The output of AD620 (1 Vpp, blue) for the input signal (100 mV, yellow)

The results of active filter AD704 for a 100 mV input signal with the gain of 10. Figures B.3 - B.8 show the results with an oscilloscope using the frequencies of 200 Hz, 300 Hz, 1 kHz, 5 kHz, 10 kHz and 12 kHz.

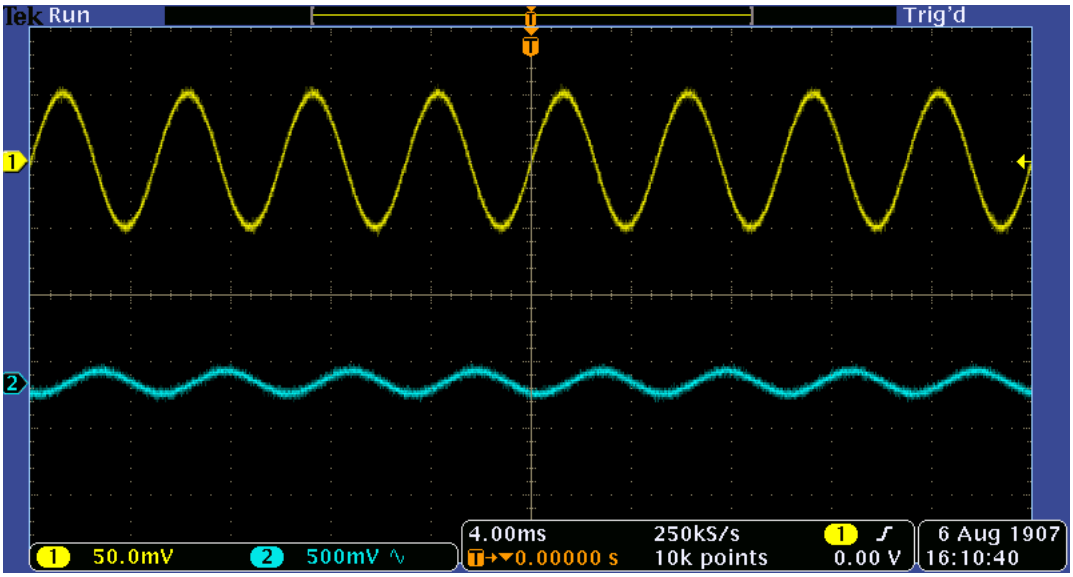


Figure B.3 Bandpass filter output is 300 mV (blue) for a 100 mV input (yellow) to the AD620 with the gain of 10. The input of the filter is 1Vpp. Frequency = 200 Hz

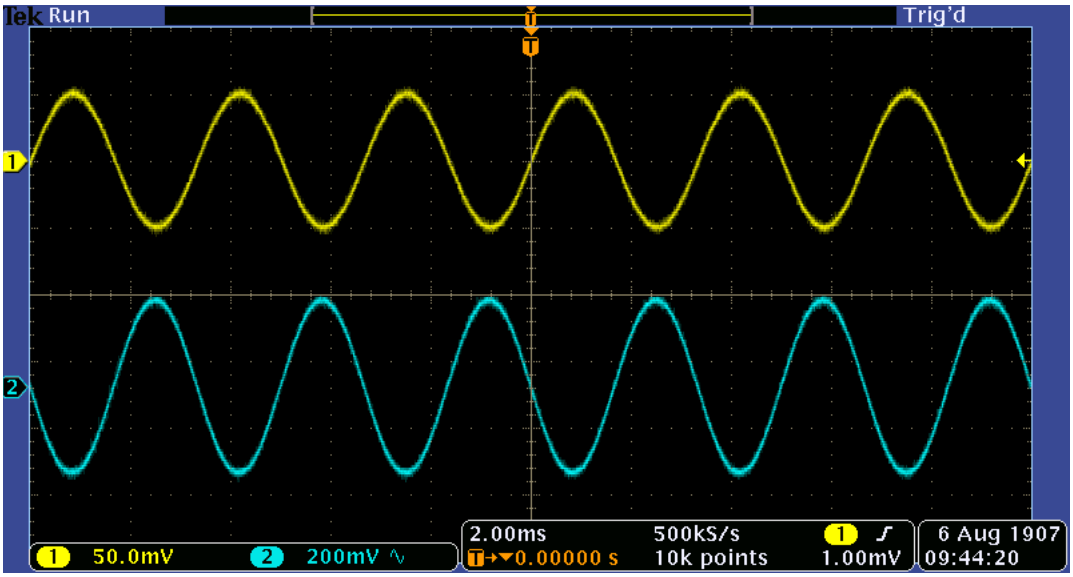


Figure B.4 Bandpass filter output is 600 mV (blue) for a 100 mV input (yellow) to the AD620 with the gain of 10. The input of the filter is 1 Vpp. Frequency = 300 Hz

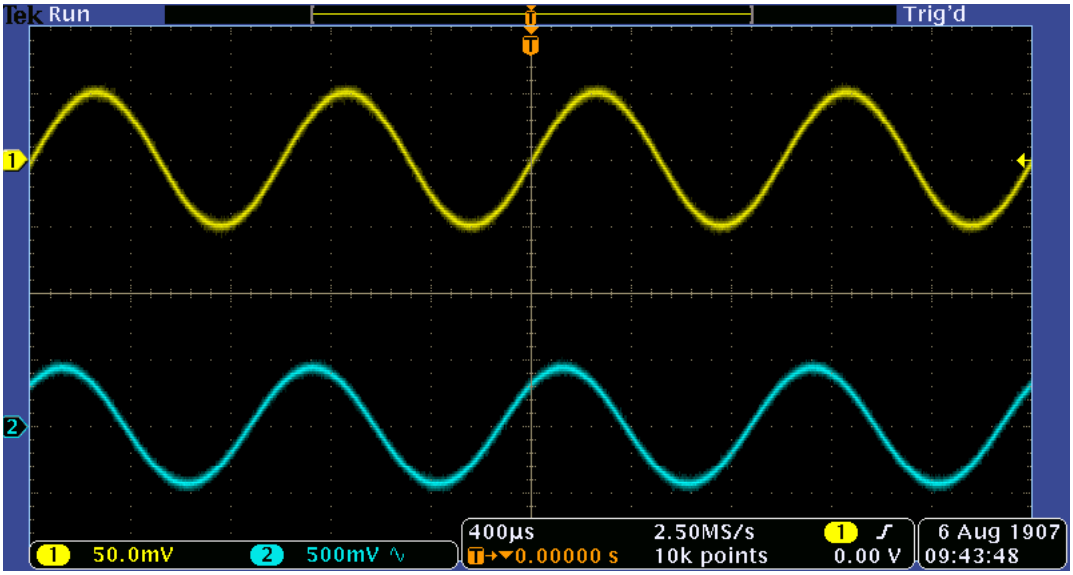


Figure B.5 Bandpass filter output is 1 V (blue) for a 100 mV input (yellow) to the AD620 with the gain of 10. The input of the filter is 1 Vpp. Frequency = 1 kHz

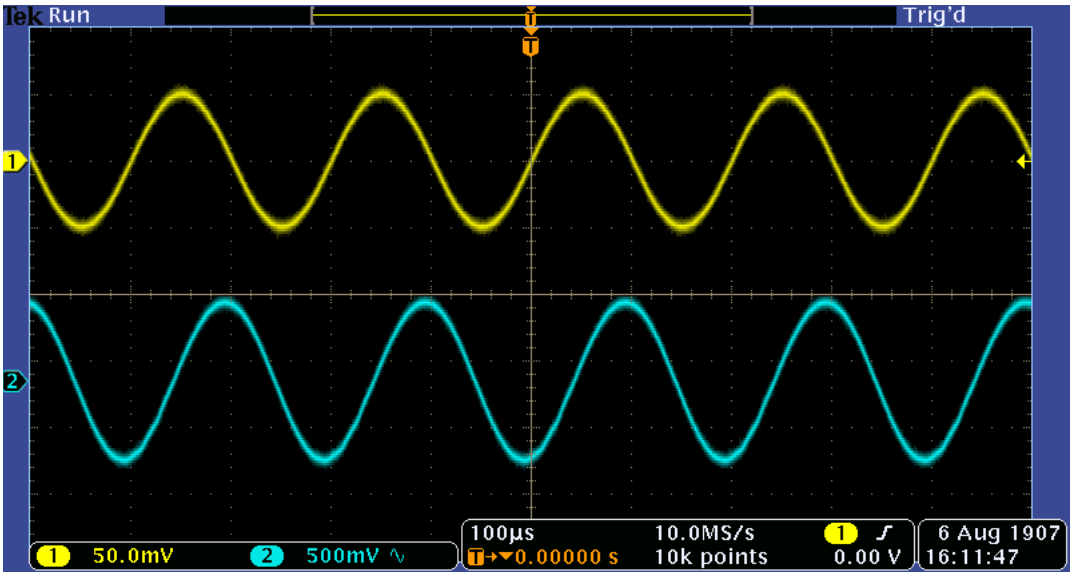


Figure B.6 Bandpass filter output is 1.25 V (blue) for a 100 mV input (yellow) to the AD620 with the gain of 10. The input of the filter is 1 Vpp. Frequency = 5 kHz

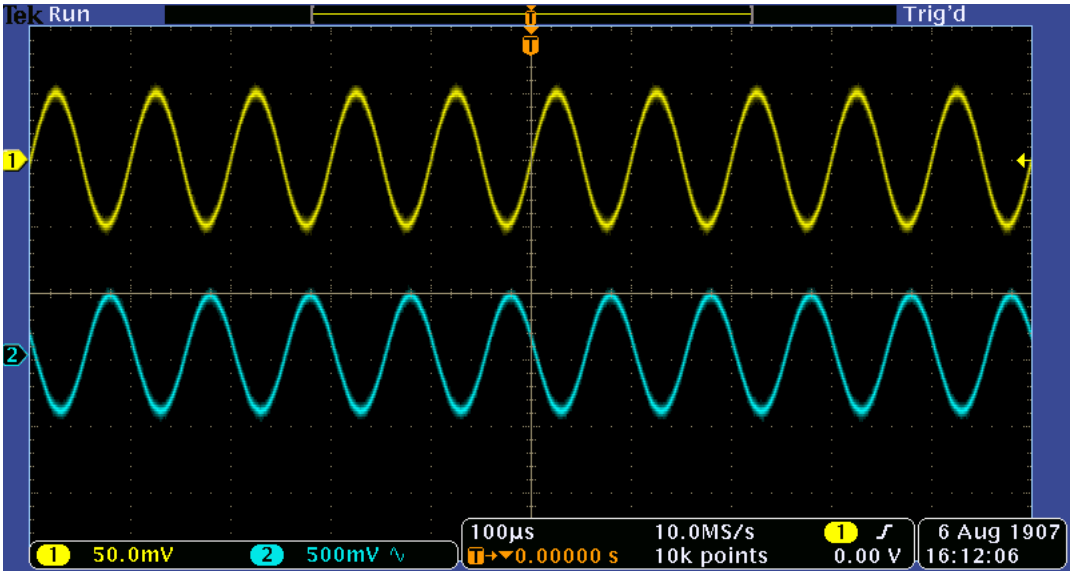


Figure B.7 Bandpass filter output is 1 V (blue) for a 100 mV input (yellow) to the AD620 with the gain of 10. The input of the filter is 1 Vpp. Frequency = 10 kHz

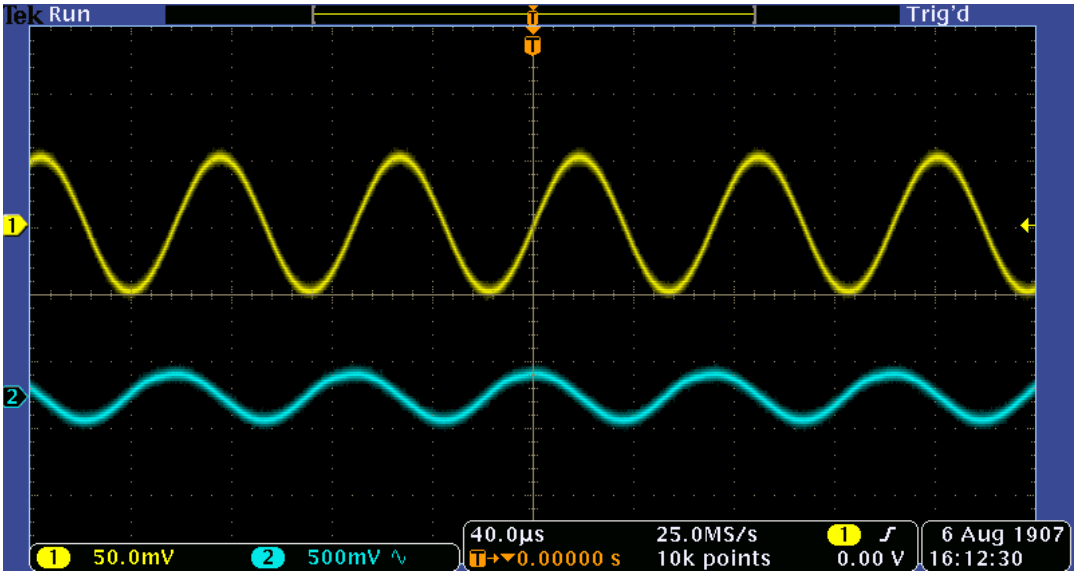


Figure B.8 Bandpass filter output is 500 mV (blue) for a 100 mV input (yellow) to the AD620 with the gain of 10. The input of the filter is 1 Vpp. Frequency = 12 kHz

The results after the ADC through SPI protocol are shown in B.9. The reference voltage is 0 to 2 V. The output is 0 and 1 V. Also, Table B.1 shows several sampling points in the binary and decimal formats.

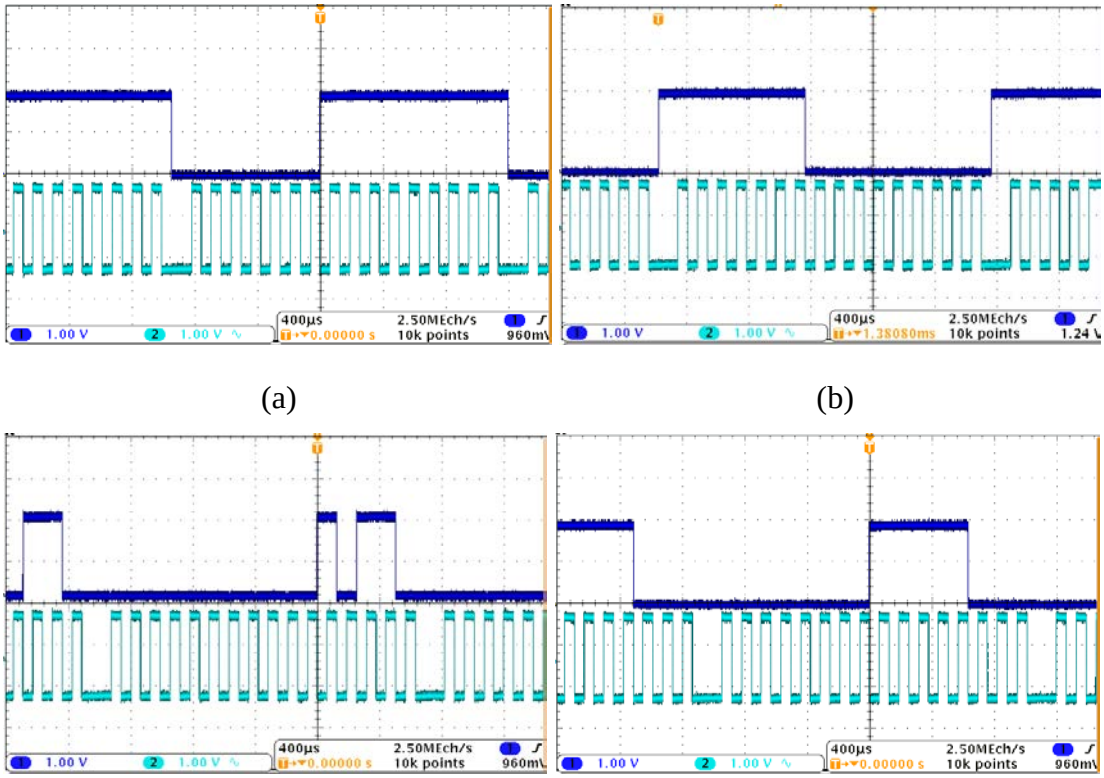


Figure B.9. The output of the ADC through SPI protocol. (a) DC Input = 0 V, 16 bits = 1111 1110 0000 0000 = -512. (b) DC Input = 2 V, 16 bits = 0000 0001 1111 1111 = 511. (c) DC Input = 1 V, 16 bits = 0000 0000 0010 1100 = 44 (d) DC Input = 1.4 V, 16 bits = 0000 0001 1111 0000 = 496

Table B.1 Some sampling voltage points in the binary and decimal formats

Samples	Binary Format	Decimal Format	Voltage
1	1111 1110 1101 0001	-303	0.49032
2	1111 1111 0111 1111	-129	0.898531
3	0000 0000 0010 0000	32	1.276243
4	0000 0000 1000 1001	137	1.522578
5	0000 0000 1001 1110	158	1.571844
6	0000 0000 0101 1000	88	1.407622
7	1111 1111 1100 1011	-53	1.07683
8	1111 1111 0001 1101	-227	0.668619
9	1111 1110 0111 1101	-387	0.293252
10	1111 1110 0001 0111	-489	0.053956
11	1111 1110 0000 0101	-507	0.011727

APPENDIX C – IMPLEMENTATION OF THE DIGITAL SIGNAL PROCESSING SYSTEM

The structure diagrams of the signal processor are shown in Figures C.1 – C.10.

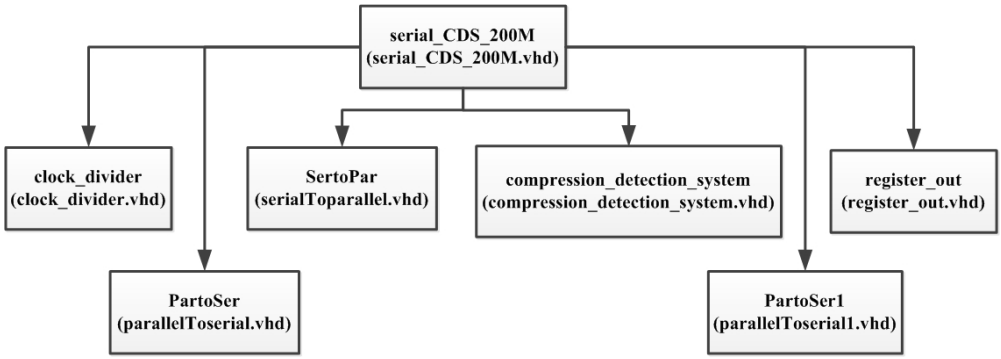


Figure C.1. The structure diagram of the serial_CDS_200M block

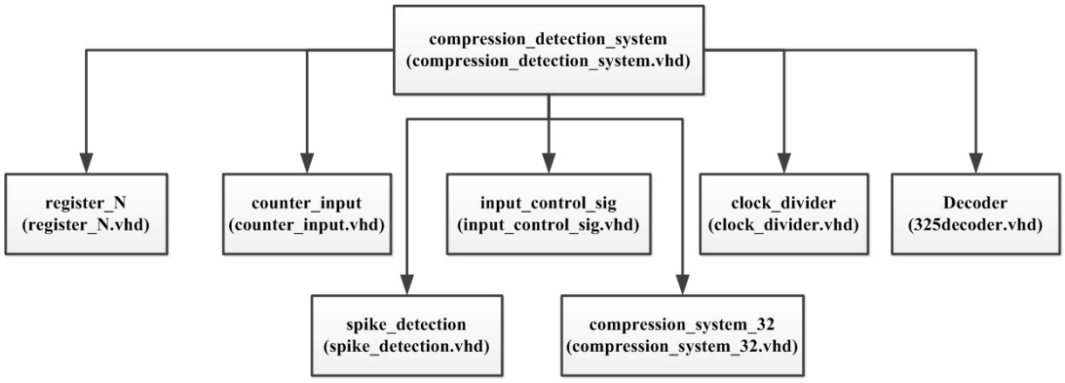


Figure C.2. The structure diagram of the compression_detection_system block

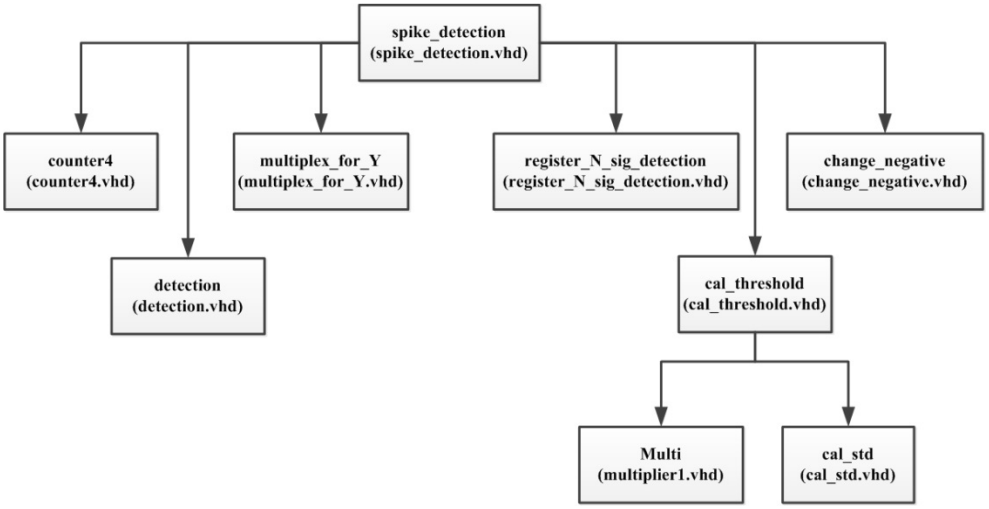


Figure C.3. The structure diagram of the spike_detection block

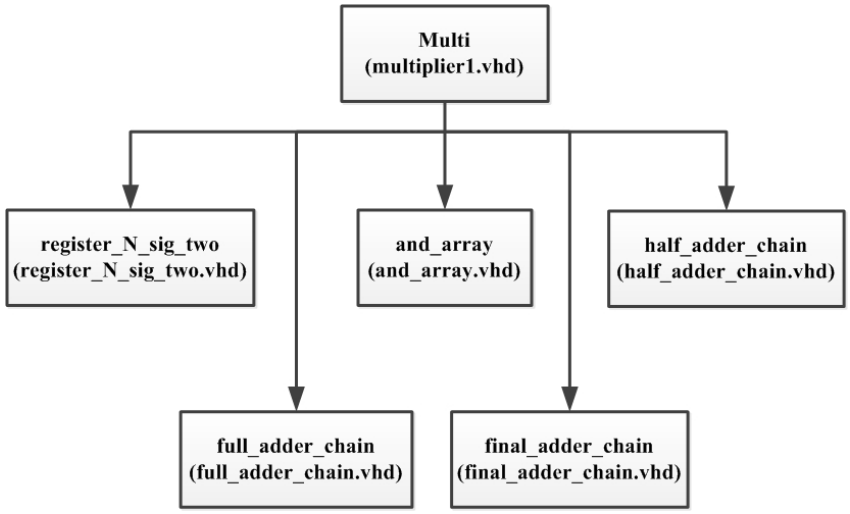


Figure C.4. The structure diagram of the multiplier1 block

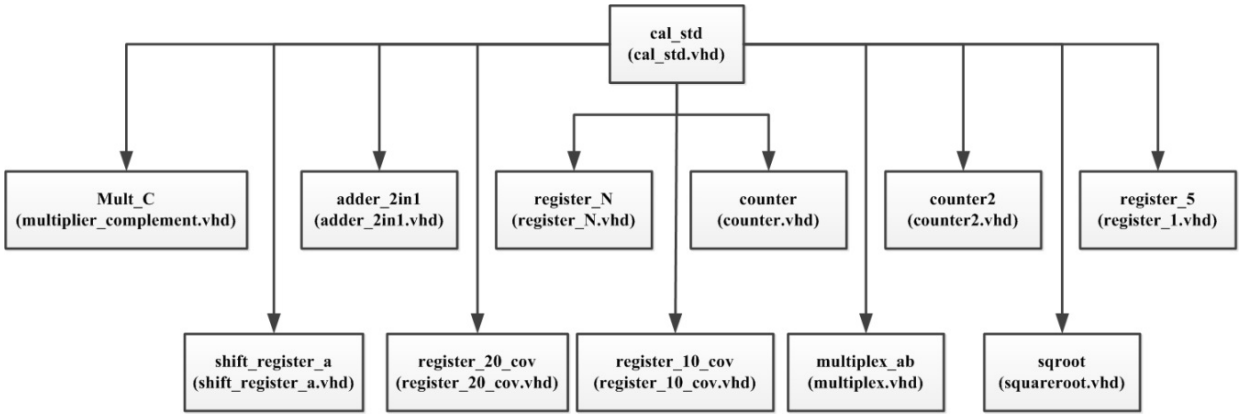


Figure C.5. The structure diagram of the standard deviation calculation block

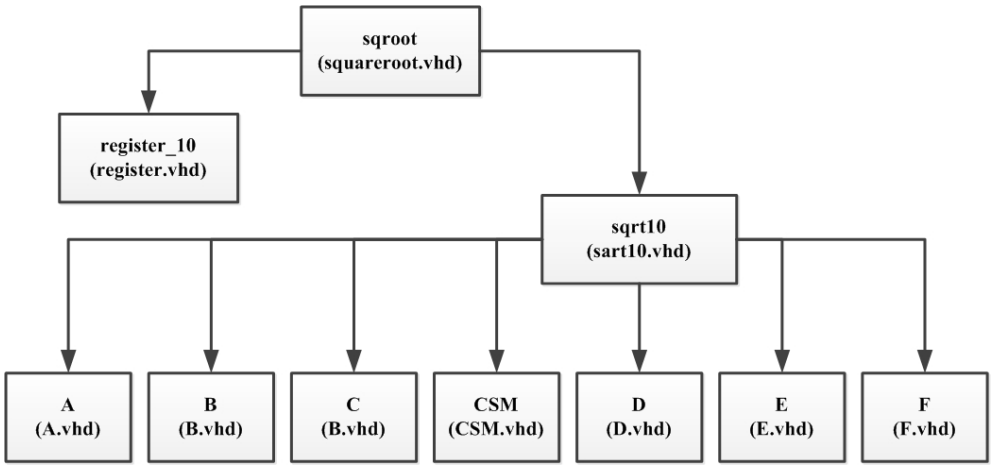


Figure C.6. The structure diagram of the square root calculation block

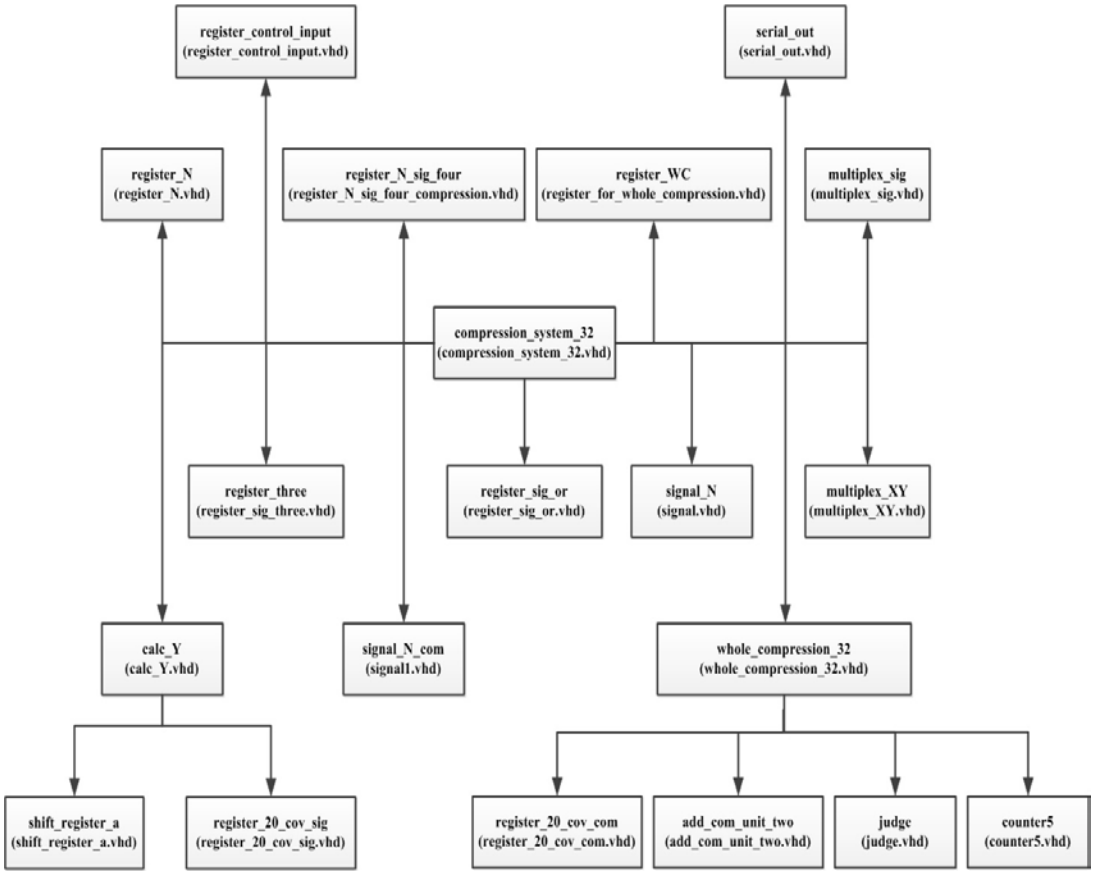


Figure C.7. The structure diagram of the data compression block

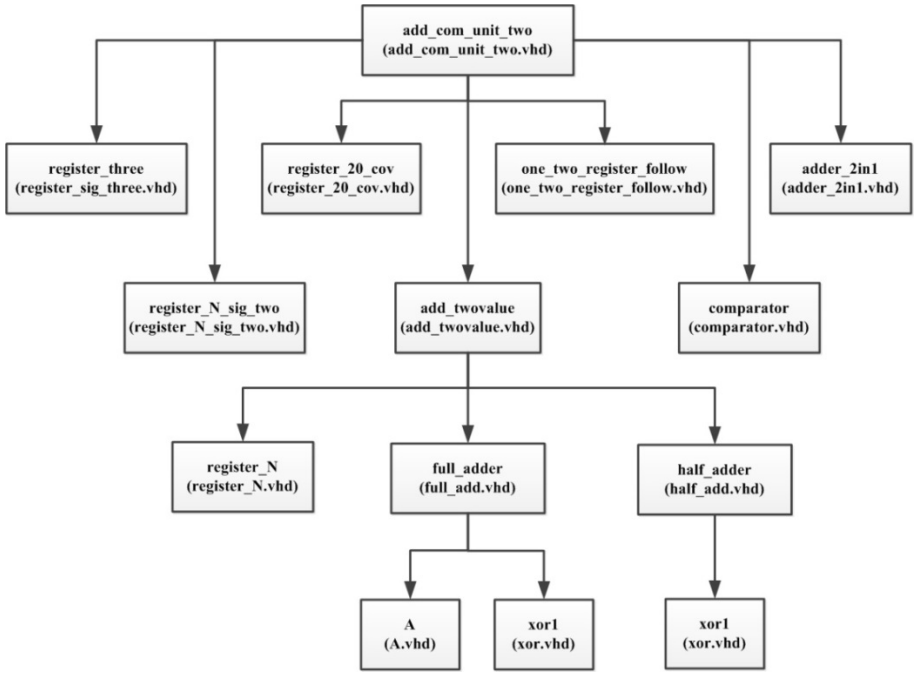


Figure C.8. The structure diagram of the add_com_unit_two block

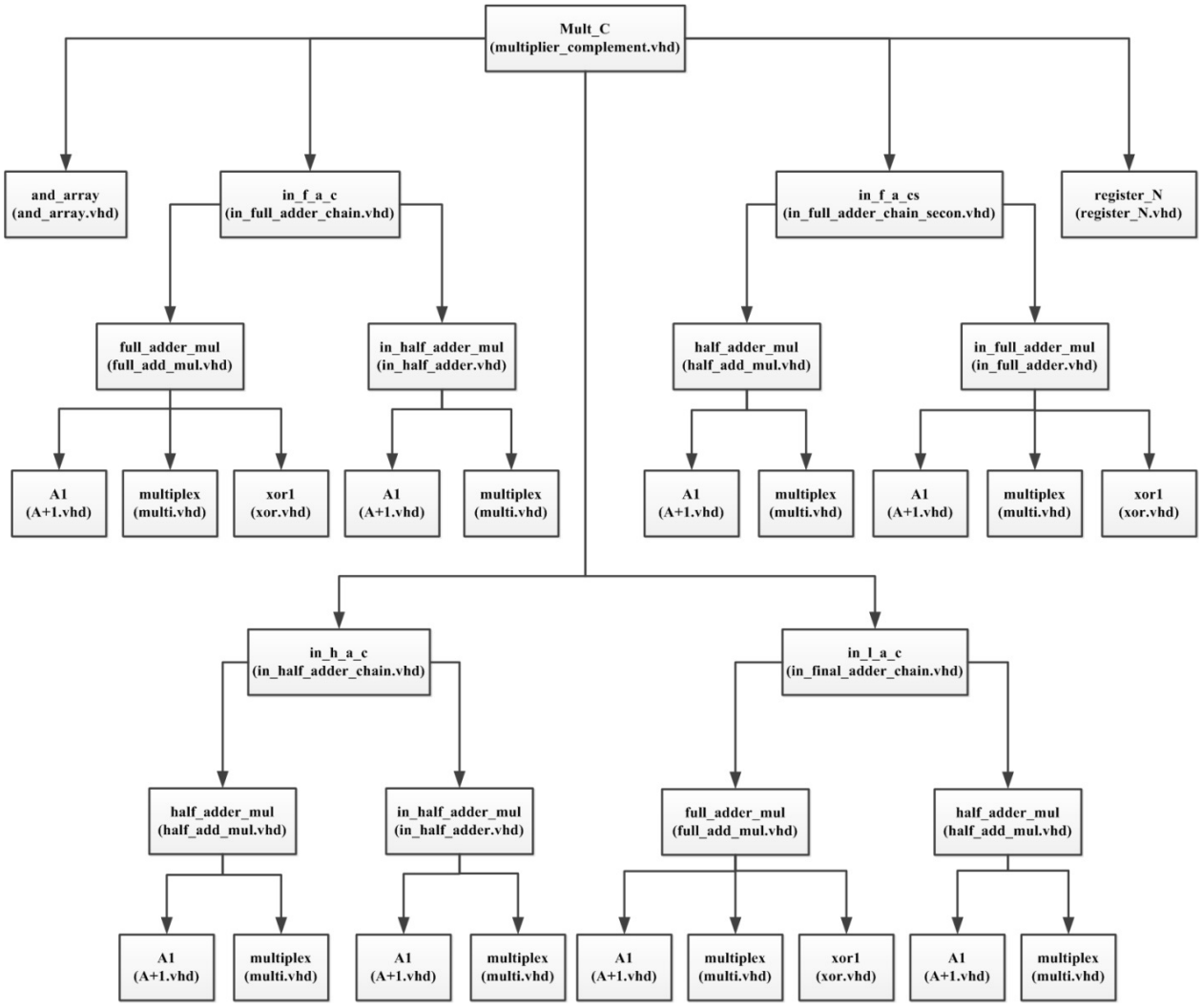


Figure C.9. The structure diagram of the Multi_C block

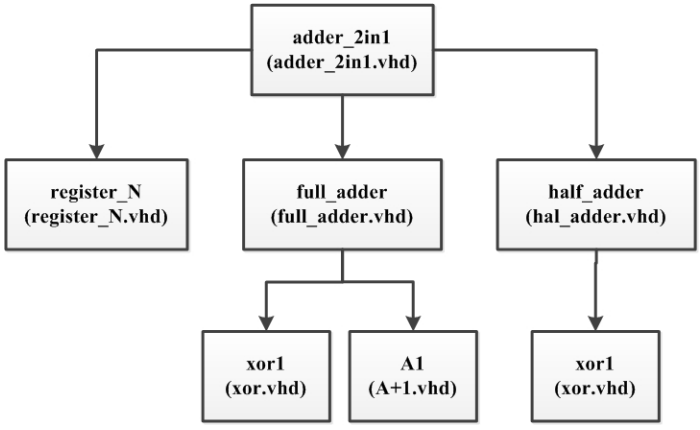


Figure C.10. The structure diagram of the adder_2in1 block

Table C.1 Explanation of some important blocks

Block name	Figure	Explanation
Serial_CDS_200M	C.1	Top module of the digital signal processing system
Clock_divider	C.1	Frequency divider
SertoPar	C.1	Serial to parallel interface
PartoSer	C.1	Parallel to serial interface
Compression_detection_system	C.1, C.2	Main module of the signal processing system
Register_N	C.2, C.5, C.7-C.10	Register
Spike_detection	C.2, C.3	Spike detection block
Compression_system_32	C.2, C.7	Data compression block
Counter,counter2,counter4, counter5, counter_input, input_control_sig register_N_sig_detection	C.2, C.3, C.5, C.7	Control signal
Multiplex_for_Y	C.3	Input of coefficient P for detection
Detection	C.3	Ouput of detection results
Change_negative	C.3	Calculation of the negative value
Cal_threshold	C.3	Calculation of the threshold
Multi	C.4	Multiplier
Cal_std	C.5	Calculation of the standard deviation
Shift_register_a	C.5, C.7	Shift register
Sqroot	C.6	Square root calculation
Whole_compression_32	C.7	Sensing matrix generation and signal compression
Serial_out	C.7	Output of the sensing matrix
Comparator	C.8	Comparator
Mult_C	C.9	Multiplier for the data using two’s complement code
Adder_2in1	C.10	Adder

Some implementations of the signal processing system in VHDL language are listed as follows.

The clock is 200M

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

------  Top module of the signal processing system, serial input and serial output ------
entity serial_CDS_200M is
generic (DATA_WIDTH : integer :=10; NUM : integer:=31; DATA_WIDTH_m : integer:=5;
```

```
DATA_WIDTH_n : integer:=5 );     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work first part
en_sam: in std_logic; -- sanpling time indication
X : in std_logic;
data_out_1: out std_logic;
data_out_2: out std_logic); -- detection begin
end serial_CDS_200M;

architecture circuits of serial_CDS_200M is
component compression_detection_system is
generic (DATA_WIDTH : integer ; NUM : integer; DATA_WIDTH_m : integer;
DATA_WIDTH_n : integer );     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in  std_logic; -- 0 is non-work, 1 is work first part
en1: in  std_logic; -- compression part
en_low: in std_logic; --0 is low_style, 1 is normal
num1: in std_logic_vector(1 downto 0);   -- num for detection counter
num2: in std_logic_vector(1 downto 0);   -- num for compression counter
num3: in std_logic_vector(1 downto 0);   -- num for compression counter divided the core
number data high power
flag_num: in std_logic_vector(2 downto 0); -- num for the detection
flag_Y: std_logic;
X : in  std_logic_vector(DATA_WIDTH-1 downto 0);
Y:  in  std_logic_vector(4 downto 0);    -- coefficient in the detection
B : in  std_logic_vector(9 downto 0);      -- core data clustering number
input_flag: out std_logic;                -- compression data begin
out_flag : out std_logic_vector(4 downto 0); -- matrix flag
P_out : out  std_logic_vector (2*DATA_WIDTH-1 downto 0); -- data
num_out: out std_logic_vector(1 downto 0); -- detection signal
TT1: out std_logic_vector(DATA_WIDTH - 1 downto 0);
TT2: out std_logic_vector(DATA_WIDTH - 1 downto 0);
flag_out: out std_logic ); -- detection begin
end component;

component SerToPar is
port(
clk : in std_logic;
rst  : in std_logic;
en: in std_logic;
serial: in std_logic;
clk_out : out std_logic;
parallel: out std_logic_vector(9 downto 0));
```

```
end component;

component clock_divider is
generic ( n : integer);
port(
clk :   in std_logic;
rst_n :  in std_logic;
clk_out: out std_logic);
end component;

component register_out is
generic (DATA_WIDTH : integer ; NUM : integer);
port(
clk : in std_logic;
rst : in std_logic;
en: in std_logic;
input_flag: in std_logic;
flag_out: in std_logic;
out_flag: in std_logic_vector(4 downto 0);
num_out: in std_logic_vector(1 downto 0);
reg_out: out std_logic_vector(8 downto 0));
end component;

component ParToSer is
generic (DATA_WIDTH : integer);
port(
clk : in std_logic;
rst : in std_logic;
en: in std_logic;
data_in: in std_logic_vector(DATA_WIDTH-1 downto 0);
data_out: out std_logic);
end component;

component ParToSer1 is
generic (DATA_WIDTH : integer);
port(
clk : in std_logic;
rst : in std_logic;
en: in std_logic;
data_in: in std_logic_vector(DATA_WIDTH-1 downto 0);
data_out: out std_logic);
end component;

signal num1, num2, num3, num_out: std_logic_vector(1 downto 0);
signal flag_num: std_logic_vector(2 downto 0);
signal out_flag: std_logic_vector(4 downto 0);
signal clk1,en_low, flag_Y, input_flag, flag_out, cout1, cout2, cout3, cout4, cout5, cout6, cout7,
```

```
rout, rout1, en_rst: std_logic;
signal max_flag: std_logic_vector(NUM-1 downto 0);
signal T11, T22, B, X1: std_logic_vector(DATA_WIDTH - 1 downto 0);
signal Y: std_logic_vector(DATA_WIDTH_n-1 downto 0); -- coefficient in the detection
signal P_out:  std_logic_vector(2*DATA_WIDTH-1 downto 0);
signal r_out:  std_logic_vector(8 downto 0);constant clk_period2 : time := 1 us;
signal AA1, enn2, enn3: std_logic;

begin
en_low <= '0';
num1 <= "11";
num2 <= "10";
num3 <= "10";
flag_num <= "010";
flag_Y <= '0';
Y <= "00000";
B <= "0000000100";

-----200MHz quartz
clkk: clock_divider
generic map(n => 1)
port map (clk => clk, rst_n => rst, clk_out => clk1);

-----50MHz quartz
clkk1: clock_divider
generic map(n => 249)
port map (clk => clk1, rst_n => rst, clk_out => cout1);

clkk2: clock_divider
generic map(n => 24)
port map (clk => clk1, rst_n => rst, clk_out => cout2);

clkk3: clock_divider
generic map(n => 1)
port map(clk => cout1, rst_n => rst, clk_out => cout4);

clkk4: clock_divider
generic map(n => 49)
port map (clk => clk1, rst_n => rst, clk_out => cout5);

clkk5: clock_divider
generic map(n => 249)
port map (clk => clk1, rst_n => rst, clk_out => cout6);

clkk6: clock_divider
generic map(n => 4)
port map (clk => clk1, rst_n => rst, clk_out => cout7);
```

```
CDS: compression_detection_system
generic map(DATA_WIDTH => DATA_WIDTH, NUM => NUM,  DATA_WIDTH_m =>
DATA_WIDTH_m, DATA_WIDTH_n => DATA_WIDTH_n)
port map(clk => cout1, rst => rst, en => en, en1 => en, en_low => en_low, num1 => num1,
num2 => num2, num3 => num3, flag_num => flag_num, flag_Y => flag_Y, X =>X1,
Y => Y, B => B, input_flag => input_flag, out_flag => out_flag, P_out => P_out, num_out =>
num_out, TT1 => T11, TT2 => T22, flag_out => flag_out);

reg: register_out
generic map(DATA_WIDTH => DATA_WIDTH, NUM => NUM)
port map (clk => cout1, rst => rst, en => en, input_flag => input_flag,flag_out => flag_out,
out_flag => out_flag, num_out => num_out, reg_out => r_out);
     --- cout4

STP: SerToPar
port map (clk => cout2, rst => rst, en => en_sam, serial => X, clk_out => cout3, parallel =>
X1);

en_rst <= not cout6 and input_flag;

PTS: ParToSer
generic map(DATA_WIDTH => 9)
port map (clk => cout5, rst => rst, en => cout4, data_in => r_out, data_out => rout);

PTS1: ParToSer1
generic map(DATA_WIDTH => 20)
port map (clk => cout7, rst => rst, en => en_rst, data_in => P_out, data_out => rout1);

data_out_1 <= rout;
data_out_2 <= rout1;
A <= AA1;
cclk <= enn3;
ss_en <= enn2;

end circuits;

----------  compression_detection_system  ----------
entity compression_detection_system is
generic (DATA_WIDTH : integer ; NUM : integer; DATA_WIDTH_m : integer;
DATA_WIDTH_n : integer );     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work first part
en1: in std_logic; -- detection part
en_low: in std_logic; --0 is low_style, 1 is normal
```

```
num1: in std_logic_vector(1 downto 0); -- num for detection counter
num2: in std_logic_vector(1 downto 0); -- num for compression counter
num3: in std_logic_vector(1 downto 0); -- num for compression counter divided the core
number data high power
flag_num: in std_logic_vector(2 downto 0); -- num for the detection
flag_Y: std_logic;
X : in  std_logic_vector(DATA_WIDTH-1 downto 0);
Y:  in  std_logic_vector(DATA_WIDTH_n-1 downto 0); -- coefficient in the detection
B : in  std_logic_vector(DATA_WIDTH-1 downto 0); -- core data clustering number
input_flag: out std_logic; -- compression data begin
out_flag : out std_logic_vector(4 downto 0); -- matrix flag
P_out : out  std_logic_vector (2*DATA_WIDTH-1 downto 0); -- data
num_out: out std_logic_vector(1 downto 0); -- detection signal
TT1: out std_logic_vector(DATA_WIDTH - 1 downto 0);
TT2: out std_logic_vector(DATA_WIDTH - 1 downto 0);
flag_out: out std_logic ); -- detection begin
end compression_detection_system;

architecture circuits of compression_detection_system is
component compression_system_32 is
generic (DATA_WIDTH : integer ; NUM : integer);     -- DATA WIDTH
port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work first part
en1: in std_logic; -- compression part
en2: in std_logic; -- controlinput
en_low: in std_logic; --0 is low_style, 1 is normal
num_in: in std_logic_vector(1 downto 0);
num_in1: in std_logic_vector(1 downto 0);
X : in std_logic_vector(DATA_WIDTH-1 downto 0);
B : in std_logic_vector(DATA_WIDTH-1 downto 0);
std_num: in std_logic_vector(4 downto 0);
std_flag: in std_logic;
input_flag: out std_logic;
out_flag : out std_logic_vector(NUM-1 downto 0);
P_out : out std_logic_vector (2*DATA_WIDTH-1 downto 0));
end component;

component spike_detection is
generic (DATA_WIDTH : integer;DATA_WIDTH_m : integer; DATA_WIDTH_n : integer );
-- DATA WIDTH
port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work
en_low: in std_logic; --0 is low_style, 1 is normal
```

```
num: in std_logic_vector(1 downto 0);
flag_num: in std_logic_vector(2 downto 0);
flag_Y: in std_logic;
X : in std_logic_vector(DATA_WIDTH-1 downto 0);
Y: in std_logic_vector(DATA_WIDTH_n-1 downto 0);
flag_std_out: out std_logic;
P_std_out : out std_logic_vector (4 downto 0);
num_out: out std_logic_vector(1 downto 0);
TT1: out std_logic_vector(DATA_WIDTH - 1 downto 0);
TT2: out std_logic_vector(DATA_WIDTH - 1 downto 0);
flag_out: out std_logic);
end component;

component register_2N_cov is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(2*DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector (DATA_WIDTH_m -1 downto 0));
end component;

component counter_input IS
generic ( DATA_WIDTH : integer);
PORT(
A: in std_logic_vector(DATA_WIDTH -1 downto 0);
CLK :IN std_logic;
rst_n :IN std_logic;
en: in std_logic;
en_inp: in std_logic;
num: in std_logic_vector (1 downto 0);
A_out: out std_logic_vector(DATA_WIDTH -1 downto 0);
CP:OUT std_logic);
END component;

component input_control_sig is
port(
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
en1: in std_logic;
en_low: in std_logic;
en_com: in std_logic;
en_int: in std_logic;
out_flag: out std_logic);
```

```
end component;

component Decoder is
port(
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work first part
din : in std_logic_vector(30 downto 0);
dout : out std_logic_vector(4 downto 0);
en_out: out std_logic);
end component;

component clock_divider IS
generic ( n : integer);
PORT(
clk :IN std_logic;
rst_n :IN std_logic;
clk_out:OUT std_logic);
end component;

component register_N is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;

signal num_cur: std_logic_vector(1 downto 0);
signal fla1, fla2, fla21,fla3, std1, std3, fla4, enout, c_out: std_logic;
signal max_flag: std_logic_vector(NUM-1 downto 0);
signal T1, X1, T11, T22, com1,com2: std_logic_vector(DATA_WIDTH - 1 downto 0);
signal T2: std_logic_vector(2*DATA_WIDTH - 1 downto 0);
signal std2, max: std_logic_vector(4 downto 0);

begin
reg: register_N
generic map(DATA_WIDTH_m => DATA_WIDTH)
port map (A=>X, clk => clk, rst_n => rst, en => en, Out_A => com1);

reg1: register_N
generic map(DATA_WIDTH_m => DATA_WIDTH)
port map (A=>X, clk => c_out, rst_n => rst, en => en, Out_A => com2);

count: counter_input
```

```
generic map(DATA_WIDTH => DATA_WIDTH)
port map (A => com1, clk =>clk, rst_n => rst, en => fla4, en_inp => std3, num => num2, A_out
=> X1, CP => std3);

count_con: input_control_sig
port map (clk =>clk, rst_n => rst, en => en, en1 => en, en_low => en_low, en_com => std1,
en_int => fla4, out_flag => fla4);

detection: spike_detection
generic map(DATA_WIDTH => DATA_WIDTH, DATA_WIDTH_m => DATA_WIDTH_m,
DATA_WIDTH_n => DATA_WIDTH_n )  -----  X => com1
port map (clk => c_out, rst => rst, en => en1, en_low => en_low, num => num1, flag_num =>
flag_num, flag_Y => flag_Y, X => com2, Y => Y, flag_std_out => std1, P_std_out => std2,
num_out => num_cur, TT1 => T11, TT2 =>T22,flag_out => fla1);

compression: compression_system_32
generic map(DATA_WIDTH => DATA_WIDTH, NUM => NUM)  --- enlager the signal enable
NUM => NUM
port map (clk =>clk, rst => rst, en => en, en1 => en, en2 => std3, en_low => en_low, num_in
=> num2, num_in1 => num3, X => X1, B => B, std_num => std2, std_flag => en, input_flag
=> fla2, out_flag => max_flag, P_out => T2); --std_flag => std1

Dec: Decoder  port map (clk =>clk, rst => rst, en => fla21, din => max_flag, dout => max,
en_out => enout);

div: clock_divider
generic map(n => 1)
port map (clk =>clk, rst_n => rst, clk_out => c_out);

fla21 <= not fla2;

input_flag <= fla2; -- compression data begin
out_flag <= max;  -- matrix flag
P_out <= T2; -- data
num_out <= num_cur; -- detection signal
flag_out <= fla1; -- detection begin
TT1 <= T11;
TT2 <= T22;

end circuits;


---------- spike detection block ----------
entity spike_detection is
generic (DATA_WIDTH : integer;DATA_WIDTH_m : integer; DATA_WIDTH_n : integer );
Port (
clk: in std_logic;
```

```
rst: in std_logic;
en: in  std_logic; -- 0 is non-work, 1 is work
en_low: in std_logic; --0 is low_style, 1 is normal
num: in std_logic_vector(1 downto 0);
flag_num: in std_logic_vector(2 downto 0);
flag_Y: in std_logic;
X : in  std_logic_vector(DATA_WIDTH-1 downto 0);
Y:  in  std_logic_vector(DATA_WIDTH_n-1 downto 0);
flag_std_out: out std_logic;
P_std_out : out  std_logic_vector (4 downto 0);
num_out: out std_logic_vector(1 downto 0);
TT1: out std_logic_vector(DATA_WIDTH - 1 downto 0);
TT2: out std_logic_vector(DATA_WIDTH - 1 downto 0);
flag_out: out std_logic);
end spike_detection;

architecture circuits of spike_detection is

component calc_threshold is
generic (DATA_WIDTH : integer;DATA_WIDTH_m : integer; DATA_WIDTH_n : integer );
Port (
clk: in std_logic;
rst: in std_logic;
en: in  std_logic; -- 0 is non-work, 1 is work
en_low: in std_logic; --0 is low_style, 1 is normal
num: in std_logic_vector (1 downto 0);
X : in std_logic_vector(DATA_WIDTH-1 downto 0);
Y:  in std_logic_vector(DATA_WIDTH_n-1 downto 0);
flag_std_out: out std_logic;
P_std_out : out std_logic_vector (4 downto 0);
flag_out: out std_logic;
P_out : out std_logic_vector (DATA_WIDTH-1 downto 0));
end component;

component counter4 IS
PORT(
CLK :IN std_logic;
rst_n :IN std_logic;
en: in std_logic;
num: in std_logic_vector (2 downto 0);
CP:OUT std_logic);
END component;

component register_N_sig_detection is
port(
sig: in std_logic;
clk: in std_logic;
```

```
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic);
end component;

component change_negative is
generic (DATA_WIDTH : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic;
A : in std_logic_vector (DATA_WIDTH-1 downto 0);
flag_out : out std_logic;
Add_out : out  std_logic_vector (DATA_WIDTH-1 downto 0));
end component;

component detection is
generic (DATA_WIDTH : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in  std_logic; -- 0 is non-work, 1 is work
en_d: in std_logic;
X : in  std_logic_vector(DATA_WIDTH-1 downto 0);
A: in std_logic_vector(DATA_WIDTH -1 downto 0);
B: in std_logic_vector(DATA_WIDTH -1 downto 0);
num_out: out std_logic_vector(1 downto 0);
flag_out: out std_logic);
end component;

component multiplex_for_Y  is
generic ( DATA_WIDTH_n : integer);
port(
A: in std_logic_vector(DATA_WIDTH_n -1 downto 0);
B: in std_logic_vector(DATA_WIDTH_n -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
en_low: in std_logic;
flag_Y: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_n -1 downto 0));
end component;

signal T1, T2: std_logic_vector(DATA_WIDTH - 1 downto 0);
signal fla, fla1, fla2, fla3, fla4, fla5, std1: std_logic;
signal sig1: std_logic_vector(DATA_WIDTH-1 downto 0);
```

```
signal sig2, Y1: std_logic_vector(DATA_WIDTH_n-1 downto 0);
signal out_num : std_logic_vector(1 downto 0);
signal std2: std_logic_vector(4 downto 0);

begin
sig1 <= (others => '1');
sig2(1 downto 0) <= "11";
sig2(DATA_WIDTH_n-1 downto 2) <= (others => '0');

count: counter4
port map (clk =>clk, rst_n => rst, en => en, num => flag_num, CP => fla);

multi: multiplex_for_Y
generic map(DATA_WIDTH_n => DATA_WIDTH_n)
port map(Y, sig2, clk, rst, en, en_low, flag_Y, fla5, Y1);

thr1: calc_threshold
generic map(DATA_WIDTH => DATA_WIDTH, DATA_WIDTH_m => DATA_WIDTH_m,
DATA_WIDTH_n => DATA_WIDTH_n)
port map (clk =>clk, rst => fla, en => fla2, en_low => en_low, num => num, X => X , Y => Y1,
flag_std_out => std1, P_std_out => std2,flag_out => fla1, P_out => T1);

thre1: register_N_sig_detection
port map (sig => fla1, clk =>clk, rst_n => rst, en => en,out_flag => fla2);

cha1: change_negative
generic map(DATA_WIDTH => DATA_WIDTH)
port map (clk =>clk, rst => fla, en => fla1, A => T1 , flag_out => fla3, Add_out => T2);

det1: detection
generic map(DATA_WIDTH => DATA_WIDTH)
port map (clk =>clk, rst => fla, en => en, en_d => fla3,  X => X, A => T1 , B => T2, num_out
=> out_num, flag_out => fla4);

num_out <= out_num;
flag_out <= fla4;
flag_std_out <= std1;
P_std_out <= std2;
TT1 <= T1;
TT2 <= T2;
end circuits;
entity spike_detection is
generic(DATA_WIDTH : integer;DATA_WIDTH_m : integer; DATA_WIDTH_n : integer );
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work
```

```
en_low: in std_logic; --0 is low_style, 1 is normal
num: in std_logic_vector(1 downto 0);
flag_num: in std_logic_vector(2 downto 0);
flag_Y: in std_logic;
X : in std_logic_vector(DATA_WIDTH-1 downto 0);
Y: in std_logic_vector(DATA_WIDTH_n-1 downto 0);
flag_std_out: out std_logic;
P_std_out : out std_logic_vector (4 downto 0);
num_out: out std_logic_vector(1 downto 0);
TT1: out std_logic_vector(DATA_WIDTH - 1 downto 0);
 TT2: out std_logic_vector(DATA_WIDTH - 1 downto 0);
flag_out: out std_logic);
end spike_detection;

architecture circuits of spike_detection is
component calc_threshold is
generic(DATA_WIDTH : integer;DATA_WIDTH_m : integer; DATA_WIDTH_n : integer );
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work
en_low: in std_logic; --0 is low_style, 1 is normal
num: in std_logic_vector (1 downto 0);
X : in std_logic_vector(DATA_WIDTH-1 downto 0);
Y: in std_logic_vector(DATA_WIDTH_n-1 downto 0);
flag_std_out: out std_logic;
P_std_out : out std_logic_vector (4 downto 0);
flag_out: out std_logic;
P_out : out std_logic_vector (DATA_WIDTH-1 downto 0));
end component;

component counter4 IS
PORT(
CLK :IN std_logic;
rst_n :IN std_logic;
en: in std_logic;
num: in std_logic_vector (2 downto 0);
CP:OUT std_logic);
END component;

component register_N_sig_detection is
port(
sig: in std_logic;
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic);
```

```
end component;

component change_negative is
generic (DATA_WIDTH : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic;
A : in std_logic_vector (DATA_WIDTH-1 downto 0);
flag_out : out std_logic;
Add_out : out std_logic_vector (DATA_WIDTH-1 downto 0));
end component;

component detection is
generic (DATA_WIDTH : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work
en_d: in std_logic;
X : in std_logic_vector(DATA_WIDTH-1 downto 0);
A: in std_logic_vector(DATA_WIDTH -1 downto 0);
B: in std_logic_vector(DATA_WIDTH -1 downto 0);
num_out: out std_logic_vector(1 downto 0);
flag_out: out std_logic);
end component;

component multiplex_for_Y  is
generic ( DATA_WIDTH_n : integer);
port(
A: in std_logic_vector(DATA_WIDTH_n -1 downto 0);
B: in std_logic_vector(DATA_WIDTH_n -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
en_low: in std_logic;
flag_Y: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_n -1 downto 0));
end component;

signal T1, T2: std_logic_vector(DATA_WIDTH - 1 downto 0);
signal fla, fla1, fla2, fla3, fla4, fla5, std1: std_logic;
signal sig1: std_logic_vector(DATA_WIDTH-1 downto 0);
signal sig2, Y1: std_logic_vector(DATA_WIDTH_n-1 downto 0);
signal out_num : std_logic_vector(1 downto 0);
signal std2: std_logic_vector(4 downto 0);
```

```
begin
sig1 <= (others => '1');
sig2(1 downto 0) <= "11";
sig2(DATA_WIDTH_n-1 downto 2) <= (others => '0');

count: counter4
port map (clk =>clk, rst_n => rst, en => en, num => flag_num, CP => fla);

multi: multiplex_for_Y
  generic map(DATA_WIDTH_n => DATA_WIDTH_n)
  port map(Y, sig2, clk, rst, en, en_low, flag_Y, fla5, Y1);

thr1: calc_threshold
generic map(DATA_WIDTH => DATA_WIDTH, DATA_WIDTH_m => DATA_WIDTH_m,
DATA_WIDTH_n => DATA_WIDTH_n)
port map (clk =>clk, rst => fla, en => fla2, en_low => en_low, num => num, X => X , Y => Y1,
flag_std_out => std1, P_std_out => std2,flag_out => fla1, P_out => T1);

thre1: register_N_sig_detection
port map (sig => fla1, clk =>clk, rst_n => rst, en => en,out_flag => fla2);

cha1: change_negative
generic map(DATA_WIDTH => DATA_WIDTH)
port map (clk =>clk, rst => fla, en => fla1, A => T1 , flag_out => fla3, Add_out => T2);

det1: detection
generic map(DATA_WIDTH => DATA_WIDTH)
port map (clk =>clk, rst => fla, en => en, en_d => fla3,  X => X, A => T1 , B => T2, num_out
=> out_num, flag_out => fla4);

num_out <= out_num;
flag_out <= fla4;
flag_std_out <= std1;
P_std_out <= std2;
TT1 <= T1;
TT2 <= T2;

end circuits;


----------  data compression block  ----------
entity compression_system_32 is
generic (DATA_WIDTH : integer ; NUM : integer);
Port (
clk: in std_logic;
rst: in std_logic;
```

```
en: in  std_logic; -- 0 is non-work, 1 is work first part
en1: in  std_logic; -- compression part
en2: in std_logic; -- controlinput
en_low: in std_logic; --0 is low_style, 1 is normal
num_in : in std_logic_vector(1 downto 0);
num_in1: in std_logic_vector(1 downto 0);
X : in  std_logic_vector(DATA_WIDTH-1 downto 0);
B : in  std_logic_vector(DATA_WIDTH-1 downto 0);
std_num: in std_logic_vector(4 downto 0);
std_flag: in std_logic;
input_flag: out std_logic;
out_flag : out std_logic_vector(NUM-1 downto 0);
P_out : out std_logic_vector (2*DATA_WIDTH-1 downto 0));
end compression_system_32;

architecture circuits of compression_system_32 is
component calc_Y is
generic (DATA_WIDTH : integer );     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work
std_num: in std_logic_vector (4 downto 0);
num1: in std_logic_vector (1 downto 0);
out_flag : out std_logic;
P_out : out std_logic_vector (DATA_WIDTH-1 downto 0));
end component;

component multiplex_XY is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
B: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
en1: in std_logic;
en_low: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;

component whole_compression_32 is
generic (DATA_WIDTH: integer; NUM : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
```

```
en: in std_logic; -- 0 is non-work, 1 is work
en_f: in std_logic;
X : in std_logic_vector(DATA_WIDTH - 1 downto 0);
Y : in std_logic_vector(DATA_WIDTH - 1 downto 0);
num_in: in std_logic_vector(1 downto 0);
flag_out: out std_logic;
flag_out_en: out std_logic_vector (NUM - 1 downto 0);
flag_out_com: out std_logic_vector (NUM -1 downto 0);
P_out0 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out1 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out2 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out3 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out4 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out5 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out6 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out7 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out8 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out9 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out10 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out11 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out12 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out13 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out14 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out15 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out16 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out17 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out18 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out19 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out20 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out21 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out22 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out23 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out24 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out25 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out26 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out27 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out28 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out29 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out30 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0));
end component;

component signal_N is
port(
A: in std_logic;
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
```

```
out_flag: out std_logic);
end component;

component counter3 IS
PORT(
CLK :IN std_logic;
rst_n :IN std_logic;
en: in std_logic;
num: out integer;
CP:OUT std_logic);
END component;

component serial_out is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
B: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
en1: in std_logic;
en2: in std_logic;
out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;

component register_N is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;

component multiplex_sig is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
B: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
en_sig: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;
```

```
component register_WC is
port(
A: in std_logic;
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic);
end component;

component register_three is
port(
A: in std_logic;
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic);
end component;

component register_N_sig_four is
port(
sig: in std_logic;
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic);
end component;

component signal_N_com is
generic (DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;

component register_sig_or is
port(
A: in std_logic;
B: in std_logic;
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic);
end component;
```

```
component register_control_input is
generic ( DATA_WIDTH : integer);
port(
A: in std_logic_vector(DATA_WIDTH -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
Out_A: out std_logic);
end component;

signal sig1, sig2, sig3: std_logic_vector(1 downto 0);
signal T1, T2: std_logic_vector(DATA_WIDTH - 1 downto 0);
signal fla, fla_or, flaa , flaa1, flaa11, fla1, fla11, fla4,fla5, fla6, fla7, fla8, clk1: std_logic;
signal fla2, fla21, fla3, sig_out1, sig_out2: std_logic_vector (NUM-1 downto 0);
type  M_array is array (0 to NUM - 1) of std_logic_vector (2*DATA_WIDTH - 1 downto 0);
signal reg_array_add : M_array;   -- addition results
signal Pout : M_array;
signal num_out : integer;

begin
std1: calc_Y
generic map(DATA_WIDTH => DATA_WIDTH)
port map (clk =>clk, rst => rst, en => std_flag, std_num => std_num, num1 => num_in1,
out_flag => fla, P_out => T1);

sig_or: register_sig_or
port map(fla, en_low, clk, rst, en1, fla_or);

sig: signal_N
port map(fla_or, clk, rst, en1, flaa);

mul1: multiplex_XY
generic map(DATA_WIDTH_m => DATA_WIDTH)
port map (A => T1, B => B, clk =>clk, rst_n => fla7, en => en1, en1 => en1, en_low =>
en_low, out_flag => fla1, Out_A=> T2); --en1 => flaa,

rci: register_control_input
generic map(DATA_WIDTH => DATA_WIDTH)
port map (A => X, clk =>clk, rst_n => rst, en => en1, Out_A => fla11);

com1: whole_compression_32
generic map(DATA_WIDTH => DATA_WIDTH, NUM => NUM)
port map (clk =>clk, rst => fla7, en => en1, en_f => fla11, X => X, Y => T2, num_in =>
num_in, flag_out => flaa1, flag_out_en => fla2, flag_out_com => fla3, --en1
P_out0 => reg_array_add(0)(2*DATA_WIDTH - 1 downto 0),
P_out1 => reg_array_add(1)(2*DATA_WIDTH - 1 downto 0),
P_out2 => reg_array_add(2)(2*DATA_WIDTH - 1 downto 0),
```

```
P_out3 => reg_array_add(3)(2*DATA_WIDTH - 1 downto 0),
P_out4 => reg_array_add(4)(2*DATA_WIDTH - 1 downto 0),
P_out5 => reg_array_add(5)(2*DATA_WIDTH - 1 downto 0),
P_out6 => reg_array_add(6)(2*DATA_WIDTH - 1 downto 0),
P_out7 => reg_array_add(7)(2*DATA_WIDTH - 1 downto 0),
P_out8 => reg_array_add(8)(2*DATA_WIDTH - 1 downto 0),
P_out9 => reg_array_add(9)(2*DATA_WIDTH - 1 downto 0),
P_out10 => reg_array_add(10)(2*DATA_WIDTH - 1 downto 0),
P_out11 => reg_array_add(11)(2*DATA_WIDTH - 1 downto 0),
P_out12 => reg_array_add(12)(2*DATA_WIDTH - 1 downto 0),
P_out13 => reg_array_add(13)(2*DATA_WIDTH - 1 downto 0),
P_out14 => reg_array_add(14)(2*DATA_WIDTH - 1 downto 0),
P_out15 => reg_array_add(15)(2*DATA_WIDTH - 1 downto 0),
P_out16 => reg_array_add(16)(2*DATA_WIDTH - 1 downto 0),
P_out17 => reg_array_add(17)(2*DATA_WIDTH - 1 downto 0),
P_out18 => reg_array_add(18)(2*DATA_WIDTH - 1 downto 0),
P_out19 => reg_array_add(19)(2*DATA_WIDTH - 1 downto 0),
P_out20 => reg_array_add(20)(2*DATA_WIDTH - 1 downto 0),
P_out21 => reg_array_add(21)(2*DATA_WIDTH - 1 downto 0),
P_out22 => reg_array_add(22)(2*DATA_WIDTH - 1 downto 0),
P_out23 => reg_array_add(23)(2*DATA_WIDTH - 1 downto 0),
P_out24 => reg_array_add(24)(2*DATA_WIDTH - 1 downto 0),
P_out25 => reg_array_add(25)(2*DATA_WIDTH - 1 downto 0),
P_out26 => reg_array_add(26)(2*DATA_WIDTH - 1 downto 0),
P_out27 => reg_array_add(27)(2*DATA_WIDTH - 1 downto 0),
P_out28 => reg_array_add(28)(2*DATA_WIDTH - 1 downto 0),
P_out29 => reg_array_add(29)(2*DATA_WIDTH - 1 downto 0),
P_out30 => reg_array_add(30)(2*DATA_WIDTH - 1 downto 0));

out_reg_199: register_N
generic map(DATA_WIDTH_m => 2*DATA_WIDTH)
port map(A => reg_array_add(30)(2*DATA_WIDTH - 1 downto 0), clk => clk, rst_n => fla4,
en => flaa1, out_A  => Pout(30)(2*DATA_WIDTH - 1 downto 0));

parti: for i in 0 to NUM - 2 generate
out_reg_i: serial_out
generic map(DATA_WIDTH_m => 2*DATA_WIDTH)
port map(A => reg_array_add(i)(2*DATA_WIDTH - 1 downto 0), B =>
Pout(i+1)(2*DATA_WIDTH - 1 downto 0), clk => clk, rst_n => fla4, en => en1,
en1=>flaa1, en2 => flaa11, out_A => Pout(i)(2*DATA_WIDTH - 1 downto 0));
end generate;

sig11: signal_N
port map(flaa1, clk, fla4, en1, flaa11);

sig12: register_three
port map(flaa11, clk, fla4, en1, fla6);
```

```
mulsig: multiplex_sig
generic map(DATA_WIDTH_m => NUM)
port map(fla21, sig_out1, clk, fla4, flaa11, fla5, fla5, sig_out1);

sig_WC: register_WC
port map(fla5, clk, rst, fla6, fla4);

sign13: register_N_sig_four
port map(flaa1, clk, rst, en1, fla7);

reg14: signal_N_com
generic map(DATA_WIDTH_m => NUM)
port map(fla2, clk, fla4, flaa1, fla21);

P_out <= Pout(0)(2*DATA_WIDTH - 1 downto 0);

input_flag <= flaa11;
out_flag <= fla3;

end circuits;


------------------  threshold calculation for spike detection  ----------
entity calc_std is
generic (DATA_WIDTH : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic;
en_low: in std_logic;
num: in std_logic_vector (1 downto 0);
X : in std_logic_vector(DATA_WIDTH-1 downto 0);
flag: out std_logic;
P_out : out std_logic_vector (4 downto 0));
end calc_std;
architecture circuits of calc_std is

component register_N is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;
```

```
component sqroot is
port ( clk: in std_logic;
rst: in std_logic;
en: in std_logic;
P : in std_logic_vector(9 downto 0);
U : out std_logic_vector(4 downto 0));
end component;

component Mult is
generic (DATA_WIDTH_m : integer; DATA_WIDTH_n : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic;
X : in std_logic_vector (DATA_WIDTH_m-1 downto 0);
Y : in std_logic_vector (DATA_WIDTH_n-1 downto 0);
P_out : out std_logic_vector (DATA_WIDTH_m + DATA_WIDTH_n -1 downto 0));
end component;

component adder_2in1 is
generic (DATA_WIDTH : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic;
A : in std_logic_vector (DATA_WIDTH-1 downto 0);
B : in std_logic_vector (DATA_WIDTH-1 downto 0);
Add_out : out  std_logic_vector (DATA_WIDTH-1 downto 0));
end component;

component counter IS
PORT(
CLK :IN std_logic;
rst_n :IN std_logic;
en: in std_logic;
num: in std_logic_vector (1 downto 0);
CP:OUT std_logic);
END component;

component counter2 IS
PORT(
CLK :IN std_logic;
rst_n :IN std_logic;
en: in std_logic;
num: in std_logic_vector (1 downto 0);
CP:OUT std_logic);
```

```
END component;

component shift_register_a is
generic (DATA_WIDTH : integer);     -- DATA WIDTH
Port (
clk : in std_logic;
rst : in std_logic;
en : in std_logic;
din : in std_logic_vector (DATA_WIDTH-1 downto 0);
num: in std_logic_vector (1 downto 0);
end component;

component Mult_C is
generic (DATA_WIDTH_m : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic;
X : in std_logic_vector (DATA_WIDTH_m-1 downto 0);
Y : in std_logic_vector (DATA_WIDTH_m-1 downto 0);
P_out : out std_logic_vector (DATA_WIDTH_m + DATA_WIDTH_m -1 downto 0));
end component;

component register_5 is
port(
A: in std_logic_vector(4 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
fla: out std_logic;
Out_A: out std_logic_vector(4 downto 0));
end component;

component register_10_cov is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector(9 downto 0));
end component;

component register_20_cov is
generic ( DATA_WIDTH : integer);
port(
```

```
A: in std_logic_vector(DATA_WIDTH -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
Out_A: out std_logic_vector(2*DATA_WIDTH-1 downto 0));
end component;

component multiplex_ab is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
B: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
en_low: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;

signal X1: std_logic_vector(2*DATA_WIDTH - 1 downto 0);
signal T1: std_logic_vector(2*DATA_WIDTH - 1 downto 0);
signal T11: std_logic_vector(2*DATA_WIDTH - 1 downto 0);
signal T12: std_logic_vector(2*DATA_WIDTH - 1 downto 0);
signal T2: std_logic_vector( 2* DATA_WIDTH - 1 downto 0);
signal T3: std_logic_vector(2* DATA_WIDTH-1 downto 0);
signal T4,T5,T51,T6, T61: std_logic_vector(2* DATA_WIDTH-1 downto 0);
signal T14, T16: std_logic_vector(DATA_WIDTH-1 downto 0);
signal T13, T15, T17: std_logic_vector(2* DATA_WIDTH-1 downto 0);

signal T7: std_logic_vector(9 downto 0);
signal T8: std_logic_vector(4 downto 0);
signal T9: std_logic_vector(4 downto 0);
signal sig4: std_logic_vector(DATA_WIDTH-1 downto 0);
signal fla, fla1, fla2, fla3, fla4, fla41,fla5, fla6, fla7: std_logic;

begin
sig4 <= (others => '1');

mul: Mult_C
generic map(DATA_WIDTH_m => DATA_WIDTH)
port map (clk =>clk, rst => rst, en => en,X => X, Y => X, P_out => T1);

add: adder_2in1
generic map(DATA_WIDTH => 2 * DATA_WIDTH)
port map (clk, rst, en, T1, T2, T2);
```

```
reg1: register_N
generic map(DATA_WIDTH_m => DATA_WIDTH * 2)
port map(T2, clk, rst, en,  T3);

count: counter port map(clk, rst, en, num, fla);

count2: counter2 port map(clk, rst, en, num, fla7);

shif: shift_register_a
generic map(DATA_WIDTH => 2*DATA_WIDTH)
port map(clk,rst,  fla,T3, num, T4);

reg2: register_N
generic map(DATA_WIDTH_m => DATA_WIDTH * 2)
port map(T4, clk, rst, en,  T5);

reg21: register_N
generic map(DATA_WIDTH_m => DATA_WIDTH * 2)
port map(T5, clk, rst, en,  T51);

count1: counter port map(clk, rst, en, num, fla1);  --en_low

con_reg: register_20_cov
generic map(DATA_WIDTH => DATA_WIDTH) --en_low
port map(X, clk, rst, en,  X1);

add1: adder_2in1
generic map(DATA_WIDTH => 2*DATA_WIDTH)
port map (clk =>clk, rst => rst, en => en, A=> X1, B => T11, Add_out => T11);

reg11: register_N
generic map(DATA_WIDTH_m => 2*DATA_WIDTH)
port map(T11, clk, rst, en,  T12);

shif1: shift_register_a
generic map(DATA_WIDTH => 2*DATA_WIDTH)
port map(clk,rst, fla1 ,T12, num,  T13);

reg12: register_10_cov
generic map(DATA_WIDTH_m => 2*DATA_WIDTH)
port map(T13, clk, rst, en,  fla4, T14); --en_low

mul1: Mult_C
generic map(DATA_WIDTH_m => DATA_WIDTH)
port map (clk =>clk, rst => rst, en => en,X => T14, Y => T14, P_out => T15);

reg13: register_10_cov
```

```
generic map(DATA_WIDTH_m => 2*DATA_WIDTH)
port map(T15, clk, rst, en,  fla41, T16);  --en_low

mul2: Mult_C
generic map(DATA_WIDTH_m => DATA_WIDTH)
 port map (clk =>clk, rst => rst, en => en, X => T16, Y => sig4, P_out => T17);

add2: adder_2in1
generic map(DATA_WIDTH => 2*DATA_WIDTH)
port map (clk =>clk, rst => rst, en => en, A => T51, B => T17, Add_out => T6);

multi1:  multiplex_ab
generic map(DATA_WIDTH_m => 2*DATA_WIDTH)
port map (A => T51, B => T6, clk =>clk, rst_n => rst, en => en, en_low => en, out_flag =>
fla3, Out_A=> T61); --en_Low

reg14: register_10_cov
generic map(DATA_WIDTH_m => DATA_WIDTH * 2)
port map(T61, clk, rst, fla3, fla5, T7);

sqr: sqroot port map(clk, rst, en, T7, T8);

reg5: register_5 port map(T8,clk, rst, fla5, fla6, T9);

P_out <= T9;
flag <= fla7;

end circuits;

---------- detection ----------
entity detection is
generic (DATA_WIDTH : integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work
en_d: in std_logic;
X : in std_logic_vector(DATA_WIDTH-1 downto 0);
A: in std_logic_vector(DATA_WIDTH -1 downto 0);
B: in std_logic_vector(DATA_WIDTH -1 downto 0);
num_out: out std_logic_vector(1 downto 0);
flag_out: out std_logic);
end detection;

architecture behavior of detection is
signal flag1: std_logic;
signal flag2: std_logic_vector(1 downto 0);
```

```
signal a1: std_logic_vector(DATA_WIDTH - 1 downto 0);

begin
a1(DATA_WIDTH - 1)<= '1';
a1(DATA_WIDTH - 2 downto 0)<= (others =>'0');

process(clk, rst, en)
begin
if(rst = '1') then
flag1 <='0';
flag2 <= "00";
else if (clk'event and clk='1') then
     if (en = '1') then
         if(en_d = '1') then
           flag1 <= '1';
           if( X(DATA_WIDTH - 1) = '0') then
             if((X > A) or(X = A)) then
             flag2 <= "01";
             else
             flag2 <= "00";
             end if;
          else if(X = a1) then
              flag2 <= "00";
              else if((X < B) or (X = B)) then
                  flag2 <= "10";
                  else
                  flag2 <= "00";
                  end if ;
              end if;
           end if;
        Else
        flag2 <= "00";
        end if;
      else
         flag2 <= "11";
         flag1 <= '0';
      end if;
   end if;
end if;
end process;

num_out <= flag2;
flag_out <= flag1;

end behavior;

------------  sensing matrix generator and data compression  --------
```

```
entity whole_compression_32 is
generic (DATA_WIDTH: integer; NUM: integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work
en_f: in std_logic;
X : in std_logic_vector(DATA_WIDTH - 1 downto 0);
Y : in  std_logic_vector(DATA_WIDTH - 1 downto 0);
num_in: in std_logic_vector(1 downto 0);
flag_out: out std_logic;
flag_out_en: out std_logic_vector (NUM - 1 downto 0);
flag_out_com: out std_logic_vector (NUM -1 downto 0);
P_out0 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out1 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out2 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out3 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out4 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out5 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out6 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out7 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out8 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out9 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out10 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out11 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out12 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out13 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out14 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out15 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out16 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out17 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out18 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out19 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out20 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out21 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out22 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out23 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out24 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out25 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out26 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out27 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out28 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out29 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out30 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0));
end whole_compression_32;

architecture circuits of whole_compression_32 is
```

```
component add_com_unit_two is
generic (DATA_WIDTH: integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work
en_f: in std_logic;
X : in std_logic_vector(2*DATA_WIDTH - 1 downto 0);
Y : in std_logic_vector(DATA_WIDTH - 1 downto 0);
flag_out1: out std_logic;
flag_out2: out std_logic;
flag_out3: out std_logic;
flag_out4: out std_logic;
P_out1 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0);
P_out2 : out std_logic_vector (2*DATA_WIDTH - 1 downto 0));
end component;

component register_20_cov_com is
generic ( DATA_WIDTH : integer);
port(
A: in std_logic_vector(DATA_WIDTH -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
Out_A: out std_logic_vector(2*DATA_WIDTH-1 downto 0));
end component;

component not_reg is
Port ( clk: in std_logic;
rst: in std_logic;
en: in std_logic;
b : in std_logic;
bo : out std_logic);
end component;


component register_N_Neg is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
en_f: in std_logic;
en_x: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
```

```
end component;
component register_N is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;

component register_Neg_follow is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;

component register_N_sig is
generic ( DATA_WIDTH_m : integer);
port(
A: in std_logic_vector(DATA_WIDTH_m -1 downto 0);
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
out_flag: out std_logic;
Out_A: out std_logic_vector(DATA_WIDTH_m -1 downto 0));
end component;

component counter5 IS
PORT(
CLK :IN std_logic;
rst_n :IN std_logic;
en: in std_logic;
num: in std_logic_vector (1 downto 0);
CP:OUT std_logic);
END component;

component judge is
generic ( NUM: integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
```

```
en: in std_logic; -- 0 is non-work, 1 is work
X : in std_logic_vector(NUM - 1 downto 0);
Out_X: out std_logic_vector(NUM-1 downto 0));
end component;

signal X1 : std_logic_vector(2*DATA_WIDTH - 1 downto 0);
signal fla: std_logic;
signal sig_a : std_logic_vector(NUM - 1 downto 0);  -- enbale signal
signal sig_b : std_logic_vector(NUM - 1 downto 0);  -- comparable signal
signal sig, sig1, sig2 : std_logic_vector(NUM - 1 downto 0);
signal sig_d : std_logic_vector(NUM - 1 downto 0);  -- addition signal
signal sig_d1 : std_logic_vector(NUM - 1 downto 0);  -- addition signal
signal sig_e : std_logic_vector(NUM - 1 downto 0);  -- enable signal
type M_array is array (0 to NUM - 1) of std_logic_vector (2*DATA_WIDTH - 1 downto 0);
signal reg_array_add : M_array;   -- addition results
signal reg_array_a : M_array;     -- first register data
signal reg_array_b : M_array;     -- second register data
signal reg_array_c : M_array;     -- thirst lever data

begin
reg: register_20_cov_com
generic map(DATA_WIDTH => DATA_WIDTH)
port map (A => X, clk =>clk, rst_n => rst, en => en, Out_A => X1);

parti: for i in 1 to NUM generate
part1 : add_com_unit_two
generic map(DATA_WIDTH => DATA_WIDTH)
port map (clk =>clk, rst => rst, en => en_f, en_f => sig(i-1), X => X1, Y => Y, flag_out1 =>
sig_a(i-1), flag_out2 => sig_b(i-1), flag_out3 => sig_e(i-1) , flag_out4 =>sig_d(i-1), P_out1 =>
reg_array_a(i-1)(2*DATA_WIDTH - 1 downto 0), P_out2 => reg_array_add(i-
1)(2*DATA_WIDTH - 1 downto 0));

end generate;
ju: judge
generic map(NUM => NUM)
port map (clk =>clk, rst => rst, en => en_f, X => sig_b, Out_X => sig);

count: counter5 port map(clk, rst, en_f, num_in, fla);

P_out0 <= reg_array_add(0)(2*DATA_WIDTH - 1 downto 0);
P_out1 <= reg_array_add(1)(2*DATA_WIDTH - 1 downto 0);
P_out2 <= reg_array_add(2)(2*DATA_WIDTH - 1 downto 0);
P_out3 <= reg_array_add(3)(2*DATA_WIDTH - 1 downto 0);
P_out4 <= reg_array_add(4)(2*DATA_WIDTH - 1 downto 0);
P_out5 <= reg_array_add(5)(2*DATA_WIDTH - 1 downto 0);
P_out6 <= reg_array_add(6)(2*DATA_WIDTH - 1 downto 0);
P_out7 <= reg_array_add(7)(2*DATA_WIDTH - 1 downto 0);
```

```
P_out8 <= reg_array_add(8)(2*DATA_WIDTH - 1 downto 0);
P_out9 <= reg_array_add(9)(2*DATA_WIDTH - 1 downto 0);
P_out10 <= reg_array_add(10)(2*DATA_WIDTH - 1 downto 0);
P_out11 <= reg_array_add(11)(2*DATA_WIDTH - 1 downto 0);
P_out12 <= reg_array_add(12)(2*DATA_WIDTH - 1 downto 0);
P_out13 <= reg_array_add(13)(2*DATA_WIDTH - 1 downto 0);
P_out14 <= reg_array_add(14)(2*DATA_WIDTH - 1 downto 0);
P_out15 <= reg_array_add(15)(2*DATA_WIDTH - 1 downto 0);
P_out16 <= reg_array_add(16)(2*DATA_WIDTH - 1 downto 0);
P_out17 <= reg_array_add(17)(2*DATA_WIDTH - 1 downto 0);
P_out18 <= reg_array_add(18)(2*DATA_WIDTH - 1 downto 0);
P_out19 <= reg_array_add(19)(2*DATA_WIDTH - 1 downto 0);
P_out20 <= reg_array_add(20)(2*DATA_WIDTH - 1 downto 0);
P_out21 <= reg_array_add(21)(2*DATA_WIDTH - 1 downto 0);
P_out22 <= reg_array_add(22)(2*DATA_WIDTH - 1 downto 0);
P_out23 <= reg_array_add(23)(2*DATA_WIDTH - 1 downto 0);
P_out24 <= reg_array_add(24)(2*DATA_WIDTH - 1 downto 0);
P_out25 <= reg_array_add(25)(2*DATA_WIDTH - 1 downto 0);
P_out26 <= reg_array_add(26)(2*DATA_WIDTH - 1 downto 0);
P_out27 <= reg_array_add(27)(2*DATA_WIDTH - 1 downto 0);
P_out28 <= reg_array_add(28)(2*DATA_WIDTH - 1 downto 0);
P_out29 <= reg_array_add(29)(2*DATA_WIDTH - 1 downto 0);
P_out30 <= reg_array_add(30)(2*DATA_WIDTH - 1 downto 0);

flag_out_en <= sig_e;
flag_out_com <= sig;
flag_out <= fla;

end circuits;

----------  Comparator for compression block ----------
entity comparator is
generic ( DATA_WIDTH : integer);
port(
clk: in std_logic;
rst_n: in std_logic;
en: in std_logic;
en_flag: in std_logic;
A: in std_logic_vector(DATA_WIDTH -1 downto 0);
B: in std_logic_vector(DATA_WIDTH -1 downto 0);
C: in std_logic_vector(DATA_WIDTH -1 downto 0);
out_flag: out std_logic;
out_flag_b: out std_logic;
Out_A: out std_logic_vector(DATA_WIDTH -1 downto 0);
Out_B: out std_logic_vector(DATA_WIDTH -1 downto 0));
end comparator;
```

```
architecture behavior of comparator is
signal Aout, Bout, zero, nul: std_logic_vector(DATA_WIDTH -1 downto 0);
signal flag, flag1: std_logic;

begin
zero <= (others => '0');
nul (DATA_WIDTH -1)<= '1';
nul (DATA_WIDTH - 2 downto 0)<= (others =>'0');

process(clk, rst_n, en)
  begin
    if(rst_n = '1') then
       Aout <= nul; --(others => '0');
       Bout <= nul; --(others => '0');
       flag <='0';
       flag1<= '0';
    else if (clk'event and clk='1') then
        if (en = '1') then
          if(A = nul) then
            flag <= '0';
            Aout <= nul;
            flag1 <= '0';
            Bout <= A;
          else if(B = zero and C= zero ) then --and en_flag = '0' ) then
              flag <= '1';
              Aout <= A;
              flag1 <= '0';
              Bout <= nul;
             else if( (B(DATA_WIDTH -1)='1' and C(DATA_WIDTH -1)='0') ) then
                 if( ((A(DATA_WIDTH -1)='0') and ((A < C) or ( A = C))) or ((A(DATA_WIDTH -
1)='1') and ((A > B) or ( A = B))) ) then
                   flag <= '1';
                    flag1 <= '0';
                    Aout <= A;
                    Bout <= nul;
                  else
                    flag <= '0';
                    flag1 <= '1';
                    Aout <= (others => '0');
                    Bout <= A;
                  end if;

            else if ( (( A > B) or ( A = B)) and ((A < C) or ( A = C))) then
                flag <= '1';
                flag1 <= '0';
                Aout <= A;
                Bout <= nul;
```

```
                else
                flag <= '0';
                flag1 <= '1';
                Aout <= (others => '0');
                Bout <= A;
                end if;
            end if;
          end if;
       end if;
        else
          flag <= '0';
          flag1 <= flag1;
          Aout <=(others => '0') ;
          Bout <= nul;
       end if;
    end if;
end if;
end process;

Out_A <= Aout;
Out_B <= Bout;
Out_flag <= flag;
Out_flag_b <= flag1;

end behavior;

----------  judger for compression block  ----------
entity judge is
generic ( NUM: integer);     -- DATA WIDTH
Port (
clk: in std_logic;
rst: in std_logic;
en: in std_logic; -- 0 is non-work, 1 is work
X : in std_logic_vector(NUM - 1 downto 0);
Out_X: out std_logic_vector(NUM-1 downto 0));
end judge;

architecture behavior of judge is
signal X1: std_logic_vector(NUM-1 downto 0);
signal nu: integer;

begin
process(clk, rst, en)
  begin
     if(rst = '1') then
        X1 <=(others => '0');
        nu <= 0;
```

```
    else if (clk'event and clk='1') then
        if(en = '1')then
          if(nu = 1) then
            for i in 0 to NUM -2 loop
              if(X(i) = '1') then
                X1(i) <= '1';
                X1(i-1 downto 0) <=(others => '0');
                X1(NUM-2 downto i+1) <=(others => '0');
                exit;
             end if;
            end loop;
            nu <= nu+1;
          else
            X1 <= (others => '0');
            nu <= nu +1;
             if(nu = 3) then
               nu <=0;
            end if;
        end if;
       else
          nu <= 0;
          X1 <= (others => '0');
       end if;
     end if;
end if;
end process;

Out_X <= X1;

end behavior;
```